НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Комп'ютерні системи та мережі»
спеціальності 123 «Комп'ютерна інженерія»
на тему: «Додаток для розпізнавання дорожніх знаків»

Виконав:
студент IV курсу, групи ІО-62
Бединський Антон Вікторович _____

Керівник:
Асистент
Регіда Павло Генадійович _____

Консультант з нормоконтролю:
Професор, доктор технічних наук
Сімоненко Валерій Павлович _____

Рецензент:
_____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Сергій СТІРЕНКО

«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Бединському Антон Вікторовичу

1. Тема проєкту «Додаток для розпізнавання дорожніх знаків», керівник проєкту Регіда Павло Генадійович, асистент, затверджені наказом по університету від «07» травня 2020 р. № 1081-с

2. Термін подання студентом проєкту 06 червня 2020 р.

3. Вихідні дані до проєкту: технічне завдання, науково-технічна література

4. Зміст пояснювальної записки: огляд існуючих рішень, вибір і обґрунтування структкури системи та розробка і тестування додатку

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) :

1. Діаграма класів додатку– плакат;
2. Метод Віола-Джонса – плакат;
3. Загальна структура системи – плакат.

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Сімоненко В.П., професор		

7. Дата видачі завдання 01 вересня 2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Затвердження теми роботи	01.09.2019-22.12.2019	
2	Вивчення та аналіз завдання	23.12.2019-22.03.2020	
3	Огляд існуючих додатків та алгоритмів	23.03.2020-01.04.2020	
4	Огляд і вибір інструментів для розробки	02.04.2020-10.04.2020	
5	Розробка детектора і класифікатора	11.04.2020-20.04.2020	
6	Розробка додатку та його тестування	21.04.2020-30.04.2020	
7	Оформлення пояснювальної записки	01.05.2020-23.05.2020	
8	Передзахист	24.05.2020-26.05.2020	
9	Захист	15.06.2020-20.06.2020	

Студент Антон БЕДИНСЬКИЙ

Керівник Павло РЕГІДА

АНОТАЦІЯ

Дана дипломна робота присвячена розробці додатку для розпізнавання дорожніх знаків. Метою розробки є додаток, який здатний в реальному часі розпізнавати дорожні знаки та інформувати користувача.

У цій роботі був проведений порівняльний аналіз існуючих на даний момент програмних рішень, які мають функцію розпізнавання дорожніх знаків. Через стрімкий розвиток комплексних систем допомоги водію, систем круїз-контролю та систем авто-пілоту, які як підсистему містять розпізнавання дорожніх знаків, було прийнято рішення створити додаток для розпізнавання дорожніх знаків.

ANNOTATION

This thesis is devoted to the development of an application for road sign recognition. The purpose of the development is an application that is able to recognize road signs in real time and inform the user.

In this work, a comparative analysis of currently existing software solutions that have the function of recognizing road signs was conducted. Due to the rapid development of advanced driver assistance systems, cruise control systems and autopilot systems, which as a subsystem include road sign recognition, it was decided to create an application for road sign recognition.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП 4671. 00.000 ВП	Відомість проєкту	1	
3	А4	ДП 4671. 01.000 ТЗ	Технічне завдання	3	
4	А4	ДП 4671. 02.000 ПЗ	Пояснювальна записка	59	
5	А4	ДП 4671. 03.000 Д1	Функціональна схема діаграми класів проєкту	1	
6	А4	ДП 4671. 04.000 Д2	Принципова схема методу Віола-Джонса	1	
7	А4	ДП 4671. 05.000 Д3	Структурна схема системи	1	
8	А4	ДП 4671. 06.000 Д4	Текст програми	17	

				ДП 4671 00.000 ВП		
	ПІБ	Підп.	Дата			
Розробн.	Бединський А.В.			Відомість дипломного проєкту	Лист	Листів
Керівн.	Регіда П.Г				1	1
Консульт.					КПІ ім. Ігоря Сікорського Каф. ОТ Гр. ІО-62	
Н/контр.	Сімоненко В.П.					
Зав.каф.	Стіренко С.Г.					

Технічне завдання
до дипломного проєкту
на тему: «Додаток для розпізнавання дорожніх знаків»

Київ – 2020 року

ЗМІСТ

					ДП 4671. 01.000 ТЗ			
Зм.	Арк.	№ докум.	Підпис	Дата				
Розробив		Бединський А.В..			Додаток для розпізнавання дорожніх знаків	Літ.	Аркуш	Аркушів
Перевір.							1	
					Технічне завдання	НТУУ “КПІ ім. Ігоря Сікорського”, ФІОТ, ІО-62		
Н. контр.		Сімоненко В.П.						
Затверд.								

НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання розповсюджується на розробку додатку для розпізнавання дорожніх знаків

Область застосування: розпізнавання дорожніх знаків для надання додаткової інформації водію

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки служить завдання на виконання роботи кваліфікаційно-освітнього рівня «бакалавр комп'ютерної інженерії», затверджене кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка Android додатку, який призначений для розпізнавання дорожніх знаків в реальному часі.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами для розробки служать науково-технічна література з інформаційних технологій, публікації в періодичних виданнях, а також відповідні статті в мережі Інтернет за даним питанням.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розроблюваного продукту

- Додаток повинен працювати в реальному часі з задовільною швидкістю;
- Додаток повинен мати достатню точність розпізнавання;

5.2. Вимоги до програмного забезпечення

- Операційна система Android 5.0 або вище
- Дозволи для використання камери та внутрішнього сховища.

5.3. Вимоги до апаратного забезпечення

- Процесор з частотою не менше 1.8 Ггц, наявність не менше 1 Гб ОЗУ
- Наявність камери та підключеної SD-карти

Пояснювальна записка
до дипломного проєкту
на тему: «Додаток для розпізнавання дорожніх знаків»

Київ – 2020 р

ЗМІСТ

Перелік термінів та скорочень

ADAS (англ. Advanced driving assistance system) Продвинута система допомоги водію.

ROI (англ. Region of interest) Область інтересу.

LBP (англ. Local binary pattern) Локальний бінарний шаблон.

CNN (англ. Convolutional neural network) Згорткова нейронна мережа.

CV (англ. Computer vision) Комп’ютерний зір.

ВСТУП

З кожним роком число автомобілів та автомобілістів збільшується, відповідно збільшується ризик ДТП на дорогах через людський чинник. В зв'язку з цим, для запобігання таких ситуацій розробляться продвинуті системи допомоги водію – ADAS(Advanced driving assistance system) та створюються системи автопілоту, які наданий момент вже вбудовуються в передові моделі машин. Такі системи збирають за допомогою датчиків,камер та радарів різноманітні данні про дорожній рух, навколишні автомобілі та інші атрибути дорожнього руху.

Одним із таких атрибутів є дорожні знаки. Вони є важлива частиною дорожньої інфраструктури є, яка надає інформації про поточний стан дороги, обмеження, заборони, попередження та іншу допомогу для інформаційного забезпечення. Нехтування або непомічення цих дорожніх знаків може прямо чи опосередковано сприяти дорожній аварії. Однак у несприятливих умовах дорожнього руху водій може випадково чи навмисно не помітити знаки руху. За таких обставин, якщо існує автоматична система виявлення та розпізнавання дорожніх знаків, вона зможе компенсувати можливу неуважність водія, допомагаючи йому слідувати маршрутом та тим самим забезпечить керування та їзду автомобілем безпечнішою.

Система розпізнавання знаків є важливим доповненням до систем допомоги водію. Основними обов'язками такої системи є надання інформації про знаки на дорозі. Така система повинна швидко реагувати на появу знаку, при цьому мати задовільну точність роботи. Роботи такої системи складається з 2 етапів – знаходження знаку на вхідному зображенні та його класифікація.

У даній дипломній роботі буде реалізовано як і локалізацію знаку, так і його класифікація. Для тестування даних систем буде розроблено Android додаток.

РОЗДІЛ 1
ОГЛЯД ІСНУЮЧИХ РІШЕННЬ

1.1. Постановка задачі

Як було сказано вище, система розпізнавання знаків - це система підтримки драйверів, яка може використовуватися для оповіщення та попередження водія у несприятливих умовах. Ця система є системою, що яка базується на областях комп'ютерного зору, яка, як правило, має можливість виявляти та розпізнавати всі знаки, навіть ті, які можуть бути частково закриті або дещо спотворені. Основними її завданнями є локалізація знака, його ідентифікація та відмінність одного знака від іншого . Алгоритм роботи такої системи зображено на рис.1.1.

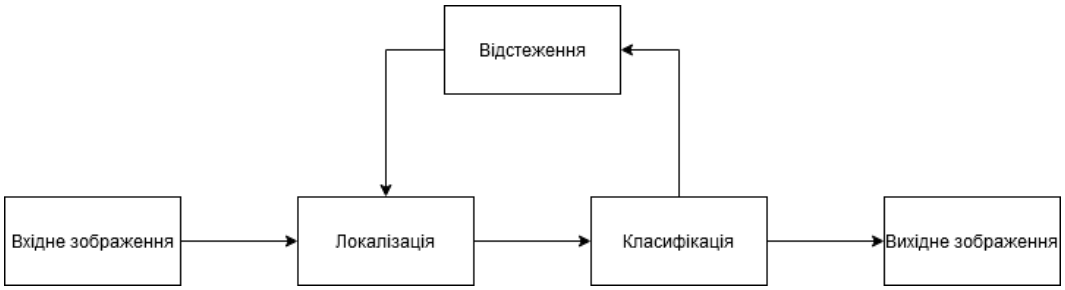


Рис.1.1. Загальна схема алгоритму роботи системи

Таким чином, система може бути розбита на дві підсистеми: система локалізації зображення та система класифікації зображення. Задача виявлення знаку – це задача пошуку знаку у вхідних даних, тоді як класифікація - це визначення того, який тип знаку було виявлено попередньою системою система . Іншими словами, виявлення трафіку знаків передбачає генерування області інтересів (ROI,region of interest), які, ймовірно, містять регіональні знаки, в той час, як класифікація означає визначання точного типу знака або відхилення запропонованої ROI і маркування цієї ROI як помилкове виявлення.

1.2. Огляд додатків, що мають функцію розпізнавання знаків:

Для того щоб розпочати розробку додатку, слід було розглянути аналоги або системи які надають можливість розпізнавання знаків. Так як цільовою

платформою розробки є Android, важливо розглянути саме додатки які працюють на даній платформі.

- **Roadly dashcam & speed camera** – Android додаток, який працює як відеорегістратор. Однією з його функцій є система розпізнавання знаків, яка розпізнає наведені знаки – “Дати дорогу”,”Заборона парковки”, “Рух заборонено”,”Пішохідний перехід”, знаки обмеження максимальної швидкості та попереджувальні знаки. Приклад розпізнавання вказано на рис 1.2.

Рис 1.2.Приклад додатку

Варто зазначити, що дані знаки належать до Російської системи дорожніх знаків. Знаки розпізнаються при 30+ кадрах в секунду, що є хорошим результатом.

- **UGV driver’s assistant** – також Android додаток, який має розпізнавання дорожніх знаків як одну з функцій. Додаток розпізнає лише попереджувальні знаки та “Пішохідний перехід”. Інтерфейс додатку показано на рис 1.3.

Рис 1.3.Приклад додатку

Одним з недоліків даного додатку є низька частота кадрів та високі системні характеристики пристрою – для використання всіх функцій потрібно не менше восьми ядер з частотою 2.0 ГГц та 2 Гб ОЗУ.

- **Tesla autopilot** – це набір продвинутих систем допомог водію, запропонованих і реалізованих компанією Tesla в своїх моделях машин, яка надає можливість адаптивного круїз-контроля, самостійного паркування, автоматичного зміни смуги та інші різноманітні можливості, інтерфейс програмно-апаратного комплексу показано на рис 1.4. Розпізнавання дорожніх знаків було введено квітні 2020 року для третьої редакції апаратного забезпечення, яка має такі характеристики – 12 процесорів ARM Cortex-A72 з тактовою частотою 2.6 МГц, 2 нейроакселератора з тактовою частотою 2 ГГц та Mali GPU з частотою 1ГГц, при цьому компанія заявляє що дана система обробляє 2300 кадрів в секунду[1]. Розпізнавання знаків використовується в режимі авто-пілота – машина реагує на знаки обмеження швидкості та зупиняється при наявності стоп-знаку. З іншого боку система не розпізнає знаки повороту, пішохідні переходи та інші. Розпізнавання виконується за допомогою нейронної мережі, але якої архітектури компанія не розкриває.

Рис 1.4. Інтерфейс Tesla Autopilot

1.3. Огляд алгоритмів та методів розпізнавання дорожніх знаків

Так як було сказано вище, що система розпізнавання знаків складається з двох підсистем, тому варто оглянути основні методи та алгоритми локалізації та класифікації зображень.

1.3.1. Огляд алгоритмів локалізації зображень

Початковим етапом в будь-якій системі розпізнавання знаків є локалізація знаку на вхідному зображенні. Так, як знаки на дорогах мають різну форму (круглі, квадратні та трикутні) та колір (червоний, синій та білий), то виводять 3 сімейства методів локалізації (рис 1.5.) – методи, які базуються на кольорах знаків, методи, які базуються на формі знаку та гібридні методи.

Рис 1.5.Методи локалізації зображень.

1.3.2. Огляд алгоритмів локалізації, що базуються на кольорі шуканого об'єкта

Методи на основі кольорів використовують контрастність знаків відносно навколишнього середовища. Ці кольори використовуються для виявлення ROI у вхідному зображенні на основі різних методів обробки зображень. Методи виявлення на основі кольорових характеристик мають низьку обчислюваність, хорошу робастність та інші характеристики, які можуть певною мірою підвищити ефективність виявлення.

Однак методи, засновані на кольоровій інформації, можна використовувати лише із набором кольорових зображень високої роздільної здатності, а також ці методи є чутливими до шумів, погодних умов та освітлення.

В методах основаних на кольорі(рис.1.6.), зображення розбивається на підмножини зв’язаних пікселей, що мають схожі кольорові характеристики. Після цього, знаки “витягуються” за допомогою сегментації порогу кольорів на основі алгоритмів. Вибір кольору є важливим для сімейства цих методів, тому існують різні методики вибору кольору – використання червоного, зеленого, синього кольорів (RGB), використання відтінку, освітленості та значення яскравості кольору (HSV) і використання відтінку, освітленості та інтенсивності (HSI).

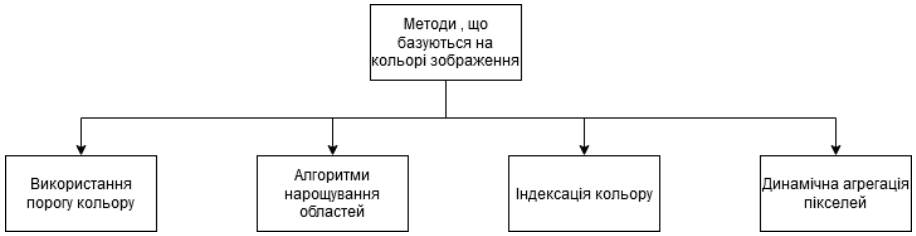


Рис.1.6.Методи, що базуються на сегментації кольору

Порогова сегментація кольору (color thresholding) одна із найдавніших технік сегментації цифрового зображення. Суть цієї техніки полягає у припущенні, що суміжні пікселі, значення яких (рівень сірого кольору, значення кольору, текстура тощо) лежить у певному діапазоні, належать до одного класу. Існує безліч різних реалізацій з використанням RGB, HSI та HSV.

Алгоритми нарощування регіонів (region growing) ще одна проста і популярна методика, яка використовується для виявлення в системах TSDR. Нарощування регіонів - це метод сегментації зображення на основі пікселів, який починається з вибору початкової точки або зернового(початкового) пікселя. Потім область розвивається шляхом додавання сусідніх пікселів, які є рівномірними, за певним критерієм відповідності, збільшуючи крок за кроком розмір області . Оскільки цей метод залежить від значень зерна, проблеми можуть виникнути, коли точки зерна лежать на краях, і, якщо в процесі росту переважають регіони, невизначеність навколо країв сусідніх областей може бути усунена неправильно.

Метод індексації кольорів - це ще один простий метод, який повністю визначає об'єкти на основі кольору. У цьому способі порівняння будь-яких двох кольорових зображень проводиться шляхом порівняння їх гістограми кольорів. Для кожної пари гістограм I та M, кожна з яких містить n проміжків, їх перетин буде визначатися за формулою (1.1):

$$\sum_{j=1}^{n} \min(I_j, M_j) \tag{1.1}$$

А значення відповідності (1.2):

$$H(I, M) = \frac{\sum_{j=1}^{n} \min(I_j, M_j)}{\sum_{j=1}^{n} M_j} \tag{1.2}$$

Перевага використання кольорових гістограм полягає в їх стійкості щодо геометричних змін проектованих об’єктів. Однак індексація кольорів залежить від сегментації, і повне, ефективне та надійне сегментування не може бути здійснене до розпізнавання. Таким чином, кольорова індексація негативно характеризується як ненадійний метод.

Інший підхід до сегментації кольорів називається **динамічною агрегацією пікселів**. У цьому способі процес сегментації здійснюється шляхом введення динамічного порогу до процесу агрегації пікселів у кольоровому просторі HSV. Застосований поріг не залежить від лінійності і його значення визначається як (1.3) :

$$a = k - \sin(s_{seed}) \tag{1.3}$$

де, k - параметр нормалізації, а S_{seed} - насиченість зернового пікселя. Основна перевага такого підходу - зменшення нестабільності відтінку. Однак даний метод не вирішує інші проблеми на основі сегментації, такі як згасання та освітлення.

1.3.3. Огляд алгоритмів локалізації, що базуються на формі шуканого об’єкта

Так само, як дорожні знаки мають конкретні кольори, вони також мають чітко визначені форми, за якими можна локалізувати зображення. Методи на

основі форми (Рис 1.7.) ігнорують колір на користь характерної форми знаків. Визначення форми є кращим для розпізнавання знаків, оскільки кольори, знайдені на знаках, змінюються залежно від освітленості. Крім того, виявлення форми зменшує пошук регіонів дорожніх знаків від усього зображення до невеликої кількості пікселів. Однак дані методи є більш ресурсоємким, що накладає обмеження на швидкодії. Пошкоджені, частково затьмарені, зів'ялі та розмиті знаки руху можуть спричинити труднощі в точному виявленні знаків руху, що призводить до зменшення точності цих методів.

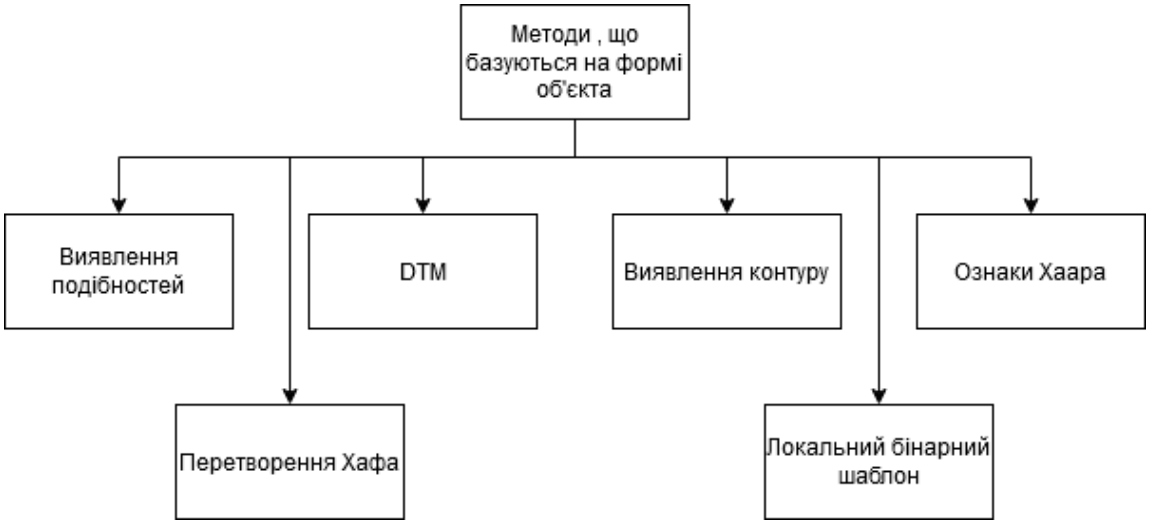


Рис.1.7. Методи, що базуються на формі об'єкта

Найпоширеніший підхід, заснований на формі, - це **перетворення Хаффа**. Перетворення Хаффа зазвичай виділяє особливості певної форми в заданому кадрі / образах . Основна перевага методики перетворення Хаффа полягає в тому, що вона терпима до прогалин в описах меж ознак і є інваріантною від шуму. Однак головним його недоліком є залежність від вхідних даних. Крім цього, він ефективний лише для великої кількості голосів, які потрапляють у правильний проміжок. Коли параметри великі, середня кількість голосів, поданих за один проміжок, стає низькою, і, таким чином, рівень виявлення зменшується.

Інший метод виявлення на основі фігури - **виявлення подібностей**. У цьому методі виявлення виконується шляхом обчислення коефіцієнта подібності між сегментованою областю та набором зразків бінаризованого

зображення, що представляють кожну форму дорожнього знаку . Головною перевагою цього методу є його прямолінійність, тоді як головний його недолік полягає в тому, що вхідне зображення повинно бути ідеально сегментоване і розміри повинні бути однаковими.

Узгодження дистанційного перетворення (DTM, Distance transform matching) - це ще один тип методу виявлення на основі фігури. У цьому способі дистанційне перетворення зображення має формуватися шляхом присвоєння кожному некрайньому пікселю значення, яке є мірою відстані до найближчого крайового пікселя.

Ця відстань обернено пропорційна відповідності зображення та шаблонам зображень. Техніка DTM є ефективною для виявлення довільних фігур у зображеннях. Однак головним його недоліком є труднощі з виявленням зашумлених і частково-закритих зображень.

Іншими популярними трьома методами виявлення знаків є **функції виявлення контуру**, **локальний бінарний шаблон (LBP)** та **Хаар-подібні особливості.** Виявлення країв відноситься до процесу ідентифікації та локалізації різких розривів в зображенні. За допомогою цього методу зображення зображень спрощуються з метою мінімізації кількості оброблюваних даних. Метод ознак Хаара був запропонований Полом Віолою та Майклом Джонсом на основі вейвлета Хаара для розпізнавання цільових об'єктів. Локальний бінарний шаблон (LBP) – метод за ідеєю доволі схожий до ознак Хаара, проте використовується локальний бінарний оператор. Основною перевагою цих трьох методів є його швидкість обчислення, де будь-який розмір зображень можна обчислити за постійний час. Однак її слабкістю є потреба у великій вибірці навчальних зображень .

1.3.4. Гібридні методи

Як обговорювалося раніше, і кольорові, і методи на основі форми мають різні переваги та недоліки. Таким чином, для ефективної локалізації знаку слід використовувати поєднання алгоритмів, що ґрунтуються на кольорі та формі. У гібридних методах будь-який колірний підхід враховується після розгляду

кольорів, або виявлення фігур використовується як основний метод, але також інтегрує деякі кольорові аспекти. У кольорових підходах зазвичай застосовується двоступенева стратегія. По-перше, сегментація дозволяє проаналізувати все зображення. Після цього застосовують методи, що локалізують по формі лише до сегментованих областей .

1.3.5. Огляд алгоритмів класифікації зображення

Після локалізації ROI використовуються методи класифікації(Рис 1.8.) для визначення вмісту виявленого об’єкту щодо певного класу дорожніх знаків . Алгоритми класифікації не ґрунтуються ні на кольорі, ні на формі. Класифікатор зазвичай приймає певний набір функцій в якості вхідних даних, що відрізняє кандидатів один від одного. Для класифікації дорожніх знаків швидко та точно використовуються різноманітні алгоритми.

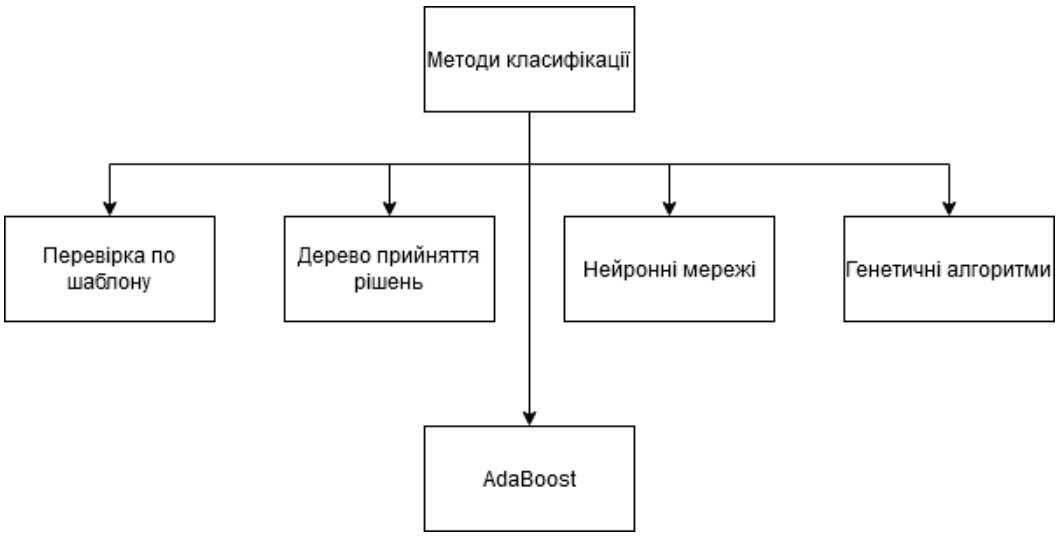


Рис.1.8. Методи класифікації

Перевірка по шаблону - це поширений метод обробки зображень та розпізнавання шаблонів. В цьому підході використовуються заздалегідь визначені шаблони для пошуку піксель за пікселем зображення за пікселем. Цей метод є доволі швидким, простим в реалізації і точним. Однак недоліком цього методу є те, що він надзвичайно чутливий до шуму. Крім цього, шаблон не є інваріантним до повороту і розміру, тобто потрібен окремий шаблон для кожного типу об’єкту, його орієнтації та розміру.

Інший розповсюджений метод – **випадковий ліс (random forest)** .Це метод машинного навчання, який під час навчання конструюються дерева прийняття рішеннь і на вихід отримується клас об’єкту.Перевагою цього методу є хороша масштабованість, розпаралелюваність, точність класифікації. До мінусів відносять великий розмір моделей, малу швидкодію в системах реального часу.

Інший популярний метод розпізнавання об'єктів - **метод глибокого навчання**. Цей метод набув загального інтересу в останні роки завдяки високій продуктивності класифікації та силі репрезентативного навчання із вихідних даних . Глибоке навчання є частиною широкого сімейства методів машинного навчання. На відміну від конкретних методів завдання, глибоке навчання фокусується на навчанні з вчителем, навчанням з підкріпленням та іншим методам. Методи глибокого навчання використовують каскад багатьох шарів нелінійних одиниць обробки для вилучення та перетворення ознак, крім цього кожен наступний шар використовує вихід із попереднього шару як вхідний. Найбільш популярними і ефективними в області класифікації зображення є згорткові нейромережі (CNN) . Існують різноманітні архітектури нейромереж, кожна з яких має свої плюси та мінуси.

Адаптивний бустінг(підсилення) або AdaBoost - це комбінація декількох алгоритмів навчання, які можна використовувати для регресії чи класифікації. Його концепція роботи заснована на побудові декількох слабких класифікаторів та їх об'єднанні в єдиний сильний класифікатор для загального завдання. Основною перевагою AdaBoost є його простота, висока потужність передбачення та можливість каскадування архітектури для підвищення обчислювальної ефективності. Однак головним його недоліком є те, що якщо вхідні дані сильно відрізняються між собою, то час навчання збільшується, а точність класифікації зменшується Крім того, тренований каскад AdaBoost не може бути динамічно скоригований з новими вхідними зразками, якщо не буде перенавчений з самого початку, що вимагає багато часу.

Доволі рідко використовуваним, але ефективним є **генетичний алгоритм.** Дане сімейство алгоритмів імітують еволюційний відбір природи, плюсами цих методів є стійкість до зміни освітлення та можливість виявлення деформованих або частково закритих знаків. Мінусами цього метода є недетермінований час роботи та негарантоване знаходження найефективнішого рішення.

ВИСНОВКИ ДО РОЗДІЛУ 1

Отже, в 1 розділі було оглянуто існуючі рішення проблеми розпізнавання знаку. Було виявлено, що на даний момент не існує відкритих систем та додатків, які надають функції розпізнавання знаку і при цьому дотримуються таких критеріїв:

- Мають велику вибірку знаків, що розпізнаються
- Мають задовільну точність та швидкодію
- Мають адекватні системні вимоги до мобільного телефону

Було виконано аналіз задачі розпізнавання знаку та розбито її на 2 підзадачі – локалізація об’єкта та його класифікація. Також було розглянуто існуючі рішення даних підзадач. Для кожного з розглянутих вище алгоритмів та методів було розглянуто їх плюси та мінуси, його особливості використання.

РОЗДІЛ 2.
ВИБІР І ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ

2.1. Опис завдання

Отже завданням на дипломну роботу є розробка Android додатку, який розпізнає дорожній знак на вхідному зображенні. Тому є доцільним вирішення таких проблем – вибір алгоритму локалізації зображення, вирішення задачі класифікації знаку, вибір програмної основи для реалізації додатку і алгоритмів та вибір даних для обраного алгоритму.

Варто зазначати що система на вхід має отримувати фото або відео з камери мобільного телефону, значить вибрані алгоритми мають мати високу швидкодію,при цьому бути точними. Додаток також має працювати на всіх сучасних телефонах на базі Android.

2.2. Вибір алгоритму локалізації зображень.

Метод Віола-Джонса(Viola-Jones)[2] – один з найперших алгоритмів, що був здатний розпізнавати образи в режимі реального часу. Хоча вперше метод був використаний для розпізнавання обличчя, його можна використовувати і для інших класів об’єктів.

Метод Віола-Джонса(Viola-Jones) має ряд принципів та особливостей які будуть розглянуті нижче.

Перший принцип - використання ознак Хаара за допомогою яких виконується пошук об’єкта на зображенні.

Ознаки Хаара – представляють собою набір прямокутних шаблонів-примітивів . В початковій реалізації(Рис 2.1.а) було використано примітиви 4 типів, але в подальшому(Рис 2.1.а) їх було розширено.

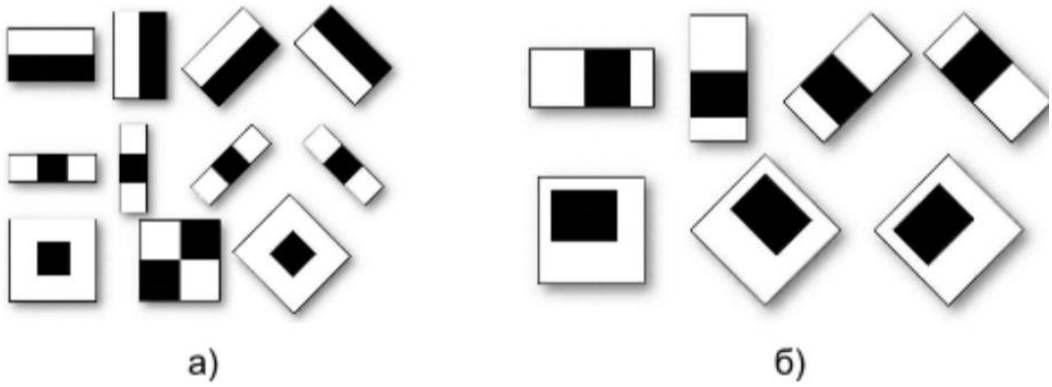


Рис 2.1.а) Ознаки Хаару б) Розширені ознаки Хаару

Значення ознаки обчислюється за формулою

$$F = X - Y \tag{2.1.}$$

Де X – сума значень пікселей закритих світлою частиною ознаки, а Y – сума значень пікселей закритих темною частиною ознаки. Кожна ознака працює в парі з пороговим значень, а рішення приймається шляхом порівняння ознаки з пороговим значенням.

Окрім цього існують модифікації алгоритму Віола-Джонса, що використовують замість ознак Хаара локальні бінарні шаблони, які мають свої плюси і мінуси в порівнянні з ознаками Хаара.

LBP(local binary pattern)[3] – локальний бінарний шаблон – це певний вид ознаки,який використовується для класифікації та локалізації в комп’ютерному зорі і представляють собою простий оператор. Дослідження показали, що LBP інваріантні до невеликих змінах в освітленості зображення або його поворот.

LBP представляє собою опис околиці пікселі в двійковому представлені, він приймає приймає центральний піксель в якості порогу. Якщо значення більше або рівне порогу, то піксель приймає значення ‘1’, якщо менший порогу, то ‘0’.Таким чином застосування оператору LBP до одного пікселя є восьмирозрядний бінарний код, який описує околицю пікселя. Приклад застосування показано на рис 2.2.

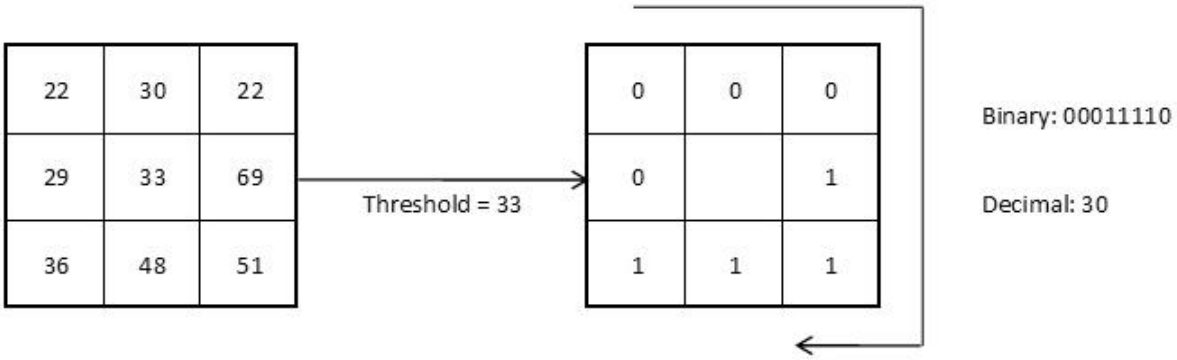


Рис 2.2.Приклад роботи оператору LBP

Також використовують розширений оператор LBP(рис 2.3.) для знаходження значень для околиць довільного розміру. Для знаходження яскравостей в точках довільної окружності пікселя використовують білінійну інтерполяцію, тобто точці присвоюють середнє зважене значення сусідніх пікселів.

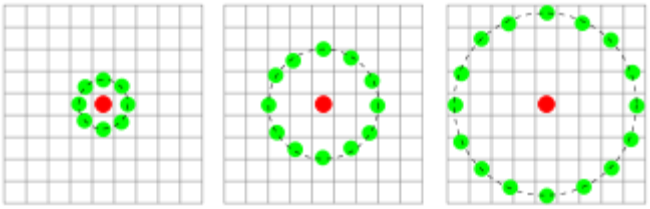


Рис 2.3.Розширений оператор LBP

Деякі бінарні коди несуть більше в собі інформації ніж інші, наприклад LBP називають рівномірним, якщо він містить не більше трьох серій ‘0’ і ‘1’ (00000000, 001110000 и 11100001) .Доцільно використовувати рівномірні LBP(рис 2.4.), адже вони виділяють такі важливі особливості зображення як кінці ліній, кути тощо.

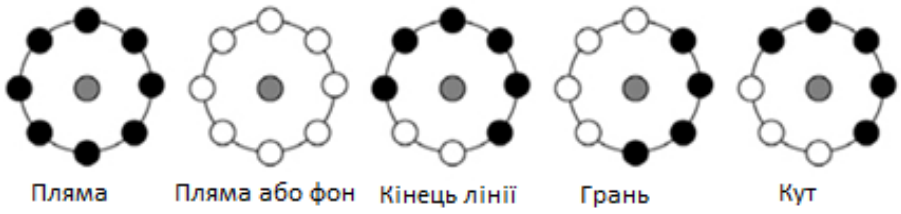


Рис 2.4.Приклади різних особливостей рівномірних LBP.

Після застосування LBP до кожного пікселя зображення, можна побудувати гістограму в якій кожному рівномірному LBP відповідає стовбець

гістограми. Для отримання загальної картини зображення варто об’єднати гістограми кожного пікселя в одну фінальну гістограми, яка буде мати як і глобальні так і локальні ознаки зображення. Приклад використання LBP показано на рис 2.5.

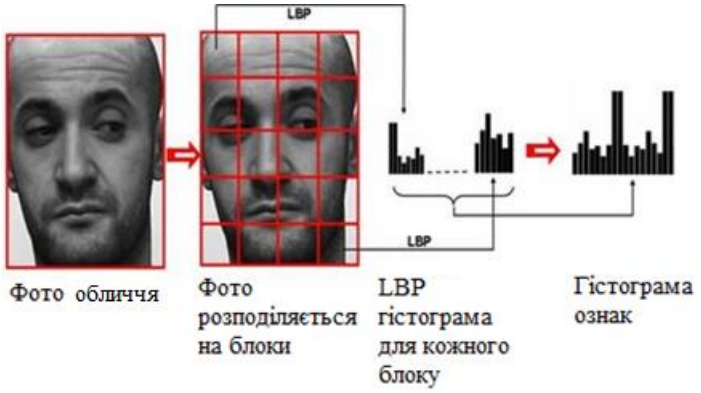


Рис 2.5.Конкатенація гістограм зображення

Другий принцип - використання інтегрального представлення зображення.

Інтегральне представлення [4] – це матриця, яка має розміри аналогічні як у початкового зображення, значення елементів якої розраховуються як сума інтенсивностей пікселів що лівіше і зверху від цього пікселя. На рис 2.6. зображено відмінності між звичайним растровим зображенням та інтегральним представленням.

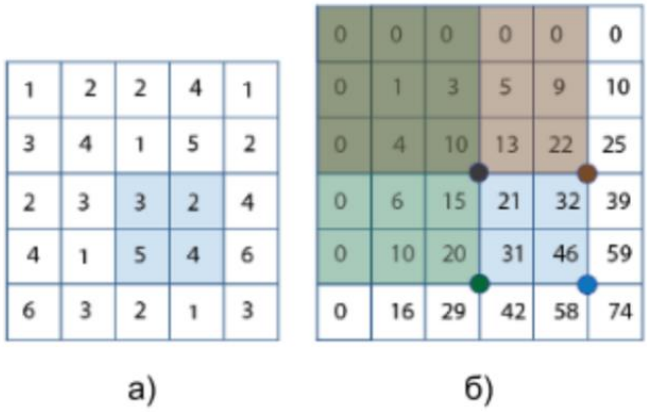


Рис 2.6.Представлення зображення а)Растрове,б) Інтегральне

Представити це можна за такою формулою:

$$I(x,y) = \sum_{\substack{x'<x \\ y'<y}} i(x',y') \qquad (2.2.)$$

Де i(x,y) значення інтенсивності пікселя у точці (x,y).По отриманій матриці інтенсивностей можна за постійний час обрахувати значення пікселей в довільному прямокутнику. Наприклад є довільний прямокутник ABCD(рис 2.7.).

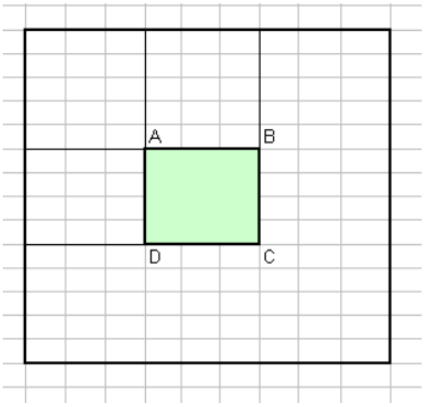


Рис 2.7.Прямокутник ABCD

Тоді площа прямокутника ABCD:

$$S(ABCD) = I(A) + I(C) - I(B) - I(D) \quad (2.3.)$$

де I(A) – інтегральне представлення в точці A.

Третім принципом є використання техніки **бустінга(boosting)**[5] – процедура послідовної побудови композиції алгоритмів машиного навчання, коли кожний наступний алгоритм намагається компенсувати недоліки попередньої композиції алгоритмів.

AdaBoost (adaptive boosting)[6]– метод бустінга, в якому кожен наступний класифікатор будується по об’єктам які погано класифікуються попередніми класифікаторами.

Математично представимо AdaBoost на прикладі бінарного класифікатора. Розглянемо задачу класифікації на два класа, Y={-1,+1}, базовий алгоритм b_n видає тільки два значення – 1 і +1 за правилом $C(b) = sign(b)$.

Шукана алгоритмічна композиція має вигляд (2.4.):

$$A(x) = C(F(b_1(x)..b_t(x))) = sign(\sum_{t=1}^{T} a_t b_t(x)) \quad (2.4.)$$

Функціонал якості композиції Q_t визначається як кількість помилок, які допускаються на тренувальній вибірці X^l:

$$Q(b, W^l) = Q_T = \sum_{t=1}^{l} w_i [y_i b(x_i) < 0] \quad (2.5.)$$

де W^l вектор ваг об'єкт. Для вирішення задачі оптимізації параметра α_t аналітично апроксимують порогову функцій втрат [z<0] за допомогою E(z) = exp(-z).

Отже, в загальний алгоритм AdaBoost виглядає так:

1.Ініціалізуються ваги об'єкта: $w_i=1/l$,i=1,..l, l – розмір тренувальної вибірки X^l

2.Для всіх t=1,..T поки не виконується критерій зупинки:

2.1.Знаходимо класифікатор b_T:X->{-1,+1} який мінімізує зважену помилку класифікації

$$b_t = argmin_b Q(b, W^l) \quad (2.6.)$$

2.2. Перераховуємо коефіцієнт зваженого голосування для алгоритма класифікації b_T:

$$\alpha_T = \frac{1}{2} \ln \frac{1-Q(b,W^l)}{Q(b,W^l)} \quad (2.7.)$$

2.3. Перераховуємо ваги об'єктів:

$$w_i = w_i \exp\left(-\alpha_T y_i b_T(x_i)\right), i = 1,..,l \quad (2.8.)$$

2.4. Нормуємо ваги об'єктів:

$$w_i = \frac{w_i}{Z_t}; i = 1,..,l; \quad (2.9.)$$

Де, Z_t – певна нормалізуюча константа

3.Повертаємо:

$$A(x) = sign(\sum_{i=1}^{T} \alpha_i b_i(x)) \quad (2.10.)$$

AdaBoost залишається одним із найпопулярніших методів бустінга, проте існують й інші різновиди такі, як LogitBoost, GentleAdaBoost, RealAdaBoost та інші.

До плюсів AdaBoost можна віднести простоту реалізації, обчислювальна складність бустінга невелика, тому складність залежить тільки від базових алгоритмів,можливість ідентифікувати об'єкти, які є шумовими викидами.

Проте даний алгоритм має і мінуси – часто буває перенавчання при наявності великої кількості шумів у вибірці, сама вибірка має бути достатньо великою і якісною.

У випадку ознак Хаара (слабкий класифікатор) AdaBoost об’єднує їх у один сильний класифікатор. Метод Віола-Джонса об’єднує кілька таких сильних класифікатор побудованих AdaBoost в єдиний каскад -композицію класифікаторів. Для навчання каскаду будується позитивна і негативна вибірка. Класифікатор на першій ступені підбирається таким чином, щоб відкинути більшість помилкових зображень і при цьому зберегти більшість позитивних зображень. На наступному етапі кількість примітивів збільшується, при цьому помилково-позитивні елементи позначаються як негативні і навчання продовжується. Тобто, наступні етапи проходять навчання так щоб виправити помилки минулих етапів. Такий підхід дозволяє швидко відкинути більшу частину помилкових об’єктів на ранніх стадіях, що зменшує кількість обчислень. На рис. 2.8. зображено приклад такого каскаду класифікаторів.

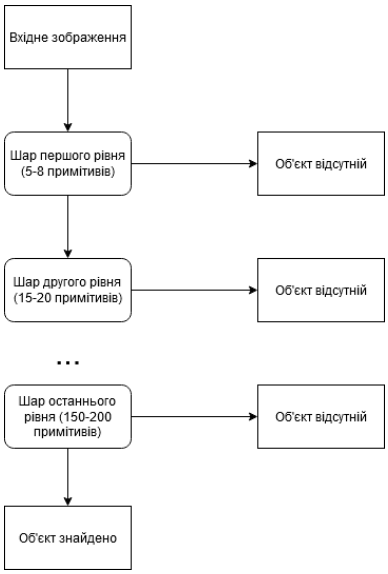


Рис 2.8. Приклад роботи каскаду

Процес детектування виконується шляхом ковзання вікна по всьому зображенню, при цьому для кожного вікна обчислюється рішення каскаду. У разі незнаходження об’єкта, розмір вікна збільшується і прохід починається знову.

Отже, у дипломній роботі доцільно буде використати метод Віола-Джонса, так як даний метод має хорошу швидкодію і точність , а також існують готові реалізації в різних бібліотеках.

2.3. Вибір алгоритму класифікації

Так як задачею класифікації в даній дипломній роботі виступає розпізнавання локалізованого знаку і віднесення його до певного класу, тому було вибрано згорткові штучні нейронні мережі, як одні із найефективніших[7] і доцільніших алгоритмів розпізнавання образів.

Згорткові нейронні мережі (Convolutional neural network, CNN) [8]– широкий клас архітектур, основна ідея полягає в тому, щоб використовувати одні і ті ж частини нейронної мережі для роботи з різними маленькими, локальними ділянками входів.

Основою CNN, як і всіх нейромереж є лінійний персептрон, вперше описаний Френком Розенблатом в 1957 році. Персептрон(рис.2.9.) складається з елементів трьох типів: S-елементи, R-елементи і A-елементи. S-елементи – шар рецепторів, ці рецептори з'єднані з A – елементами за допомогою гальмівних або збуджуючих зв'язків. A-елемент є суматором з порогом, тобто цей елемент збуджується, якщо алгебраїчна сума збуджень на вході перевищує деяку величину – поріг. При цьому, якщо сигнал приходить по гальмівному зв'язку сигнал від елементу буде від'ємним, а якщо по збуджуючому зв'язку – буде додатнім. Сигнали від збудженого A-елементу передаються в зважений суматор R.

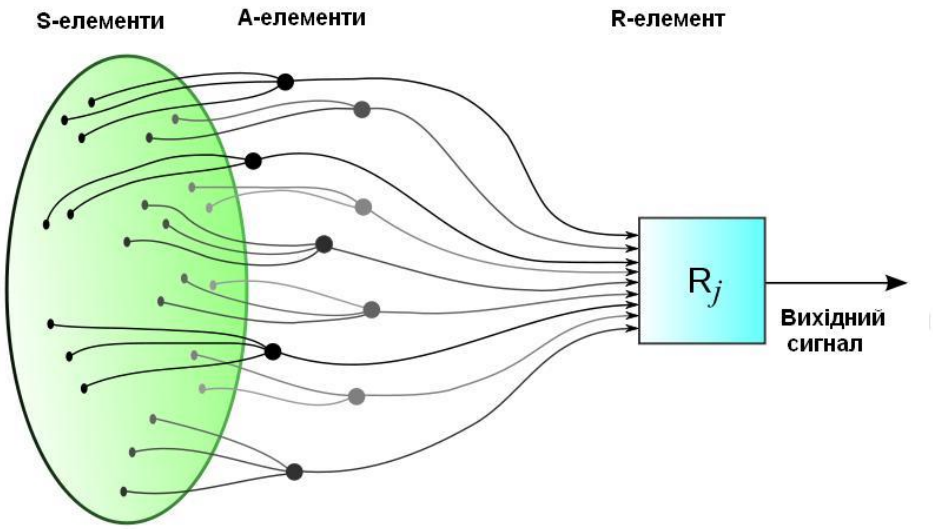


Рис 2.9.Модель персептрона

Математично персептрон можна записати так:

$$f(x) = sign(\sum_{i=1}^{n} w_i x_i - \theta) \qquad (2.11.)$$

Згорткові нейронні мережі також мають ряд особливостей, а саме:

- На кожен нейрон подається лише певна частина зображення (або виходи попереднього шару).Це дозволяє кожному нейрону вловити особливості зображення, що значно збільшує точність зображення. При цьому значно зменшується об’єм обчислень.
- Використовується концепція роздільних ваг – для великої кількості ваг використовується невелика кількість навчаємих параметрів – вагових коефіцієнтів. Це досягається з рахунок того, що нейрони однієї карти мають одні й ті самі вагові коефіцієнти. Використання такої концепції дозволяє значно пришвидшити навчання мережі.
- Використовується субдискретизація – зменшення розміру вхідного зображення без втрати значущих ознак.

Структура згорткових нейромереж також доволі відрізняються від звичайних нейромереж наявністю шарів згортки та шарів субдискретизації, які чередуються між собою.

Згортковий шар (convolutional layer) – основний будівельний шар, у якому виконується операції згортки над зображенням. Кожен такий шар складається з набору фільтрів або ядер, які мають рецептивне поле. У результаті прямого проходу кожен фільтр здійснює згортку за шириною та висотою вхідної ємності – лінійне перетворення даних особливого виду. Якщо x^l -карта ознак в шарі l, то результат двовимірної згортки з ядром розміром (2d+1) і матрицею ваг W (2d+1) x (2d+1) можна представити такими чином:

$$y^l_{i,j} = \sum_{-d \le a,b \le d} W_{a,b} x^l_{i+a,j+b} \qquad (2.12.)$$

де $y^l_{i,j}$ – результат згортки на рівні l, а $x^l_{i,j}$ – її вхід, тобто вихід попереднього шару. На рис. 2.10. і рис. 2.11. показано приклад виконання і результат операції згортки , де '*' це знак операції згортки, а не множення.

$$\begin{pmatrix} 0 & 1 & 2 & 1 & 0 \\ 4 & 1 & 0 & 1 & 0 \\ 2 & 0 & 1 & 1 & 1 \\ 1 & 2 & 3 & 1 & 0 \\ 0 & 4 & 3 & 2 & 0 \end{pmatrix} * \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 9 & 5 & 4 \\ 8 & 8 & 10 \\ 8 & 15 & 12 \end{pmatrix}.$$

Рис 2.10.Виконання операції згортки

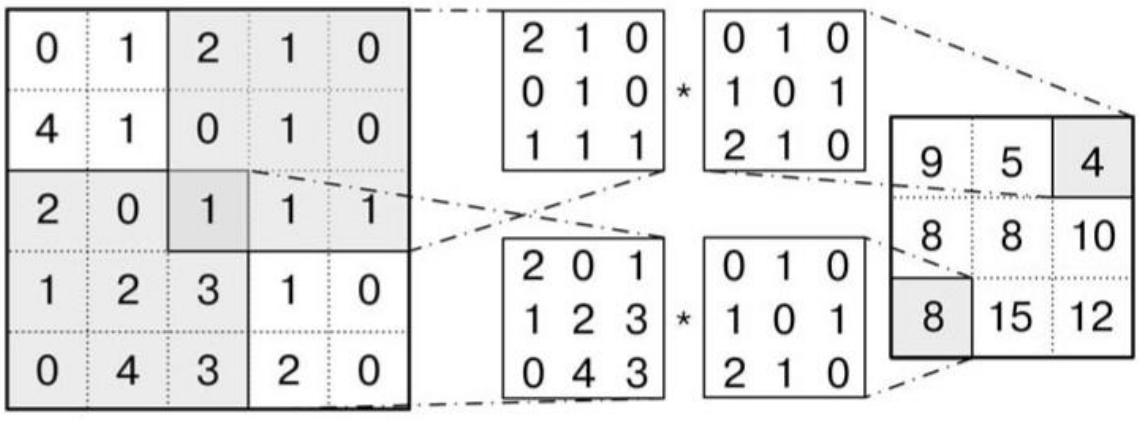


Рис 2.11.Результат виконання дій

Субдискретизуючий шар (subsampling layer) - у даному шарі зменшується розмірність отриманих карт ознак за допомогою операції субдискретизації , приклад зображено на рис 2.12. Її суть полягає у виборі із невеликої квадратної області нейронів карти ознак максимального нейрона,

далі цей нейрон приймається за один нейрон карти ознак цього шару. Це не тільки пришвидшує обчислення, але і дозволяє зробити нейромережу більш незалежною від розміру зображень.

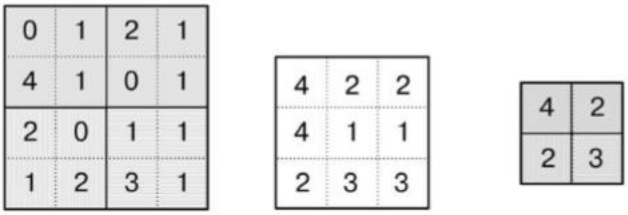


Рис 2.12. Операція субдискретизації для зображення з кроком 2 і 3.

Повнозв'язний шар (full conection layer), даний шар представляє собою персептрон, який добре розпізнає прості об'єкти отримані після перетворень у попередні двох шарах.

Для навчання згорткових нейромереж використовують алгоритми навчання з вчителем. Саме ж навчання є нічим ні іншим,як задачею оптимізації. Для оцінки навчання використовують функцію втрат – вона порівнює значені на виході з очікуваними значеннями і видає певне значення (оцінку) . Вірно обрана функція втрат може значно пришвидшити навчання мережі, в той же час погано підібрано функція може як і швидко перетренувати мережу, так і збільшити час тренування. До найпопулярніших функцій помилок належать метод найменших квадратів, метод перехресної ентропії та інші методи.

Одним із найпоширеніших методів навчання є метод зворотнього поширення помилки[9]. Його суть полягає у використанні градієнтного спуску, а саме його стохастичну модифікації для оновлення ваг нейронів. Суть цього метода полягає в русі проти градієнту(для досягнення мінімуму функції втрат) в просторі ваг нейронів і при цьому після кожного навчального прикладу оновлюються ваги нейронів.

Одним із найважливіших аспектів навчання нейромережі є функція активації нейронів, яка дозволяє досягнути нелінійності. Від її вибору залежить як і швидкість навчання, так і якість розпізнавання. Тому варто розглянути найпопулярніші функції активації:

1. Сигмоїда – математично задається формулою (2.13.) , а графік функції зображено на рис.2.13.:

$$f(x) = \frac{1}{1 + e^{-x}} \qquad (2.13.)$$

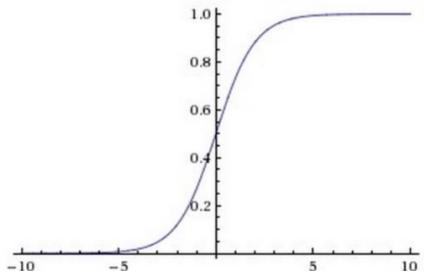


Рис 2.13.Графік сигмоїди

На вхід функція приймає будь-яке дійсне число, а видає дійсне число з проміжку (0,1), що дозволяє легко інтерпретувати рівень активації нейрона : 0 – відсутність активації, 1- повністю насичена активація. Однак дана функція активація має серйозні недоліки - при наближенні значення функції до 0 або 1,її градієнт стає близькими до 0, що призводить до ‘відмирання’ нейрона або значного зменшення швидкості його навчання.

2. Гіперболічний тангенс - математично задається формулою (2.14.) , а графік функції зображено на рис.2.14.:

$$f(x) = \frac{2}{1 + e^{-2x}} - 1 \qquad (2.14.)$$

Функція доволі схожа до сигмоїди, однак її вихід центрований відносно нуля і градієнт більше ніж у сигмоїди (похідна крутіша).

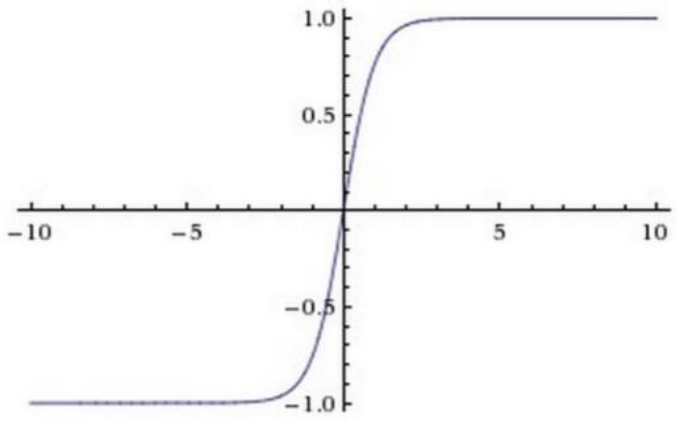


Рис 2.14. Графік гіперболічного тангенса.

Але як і сигмоїда, функція має проблеми із затуханням градієнту, що негативно впливає на навчання нейронів.

3. Випрямлена лінійна функція активації, ReLU (rectified linear unit) - математично задається формулою (2.15.) , а графік функції зображено на рис.2.15.:

$$f(x) = \max(0, x) \qquad (2.15.)$$

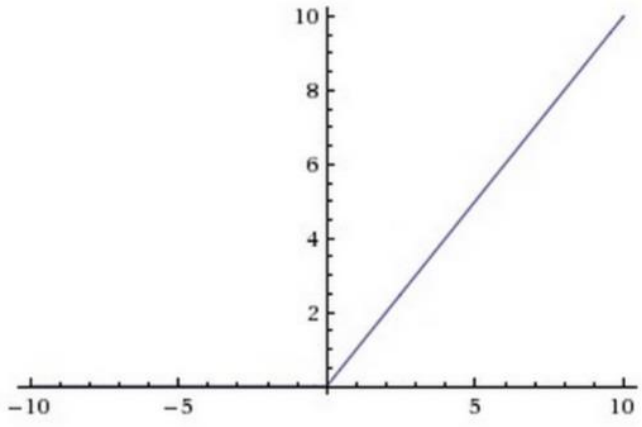


Рис 2.15.Графік ReLU

Одна із найпопулярніших за останні роки функція, яка немає недоліків властивих сигмоїді і гіперболічному тангенсі. Крім того, дана функція має меншу обчислювальну складність, що економить час навчання, та використання цієї функції призводить до пришвидшення сходимості градієнтного спуску. Це пов’язують з лінійим характером функції.

2.4. Огляд та обґрунтування вибору програмних засобів для виконання поставленої задачі

2.4.1. Вибір мови програмування

Так як буде вирішено ряд доволі різних задач – підготовка навчальних вибірок, навчання каскадів, навчання нейронної мережі та створення Android додатку було вирішено використати для створення Android додатку використати Java, а для інших задач використати Python. Це зумовленим тим, що Python не надає стабільних та швидкодійних засобів створення додатків на Android-платформу.

Для вирішення утилітарних задач було обрано Python. Python – інтерпретована мультипарадигмова об’єктно-орієнтована мова програмувння, яка розроблена в 1990 році Гвідо ван Россумом. Вона є ідеальною мовою програмування для швидкого створення і розробки різного роду скриптів через свою простоту синтаксису і велику кількість різнопланових сторонніх бібліотек.

Python є вільно розповсюджуваною мовою, яку можна скачати з офіційного сайту. Велика стандартна бібліотека є однією із сильних сторін мови – наприклад в наявності є засоби для роботи з мережевими протоколами, XML, регулярними виразами,мультимедійними форматами та інші. Крім того, існує велика кількість сторонніх бібліотек посилання на які можна знайти на офіційному сайті. Також існує можливість додавання сторонніх функцій і типів реалізованих на C і C++ в інтерпретатор Python.

Головним критерієм вибору було те що, Python є одним із провідних інструментів в області комп’ютерного зору, машинного навчання та математичних обчисленнь. Далі буде розглянуто ці бібліотеки,їхні плюси та мінуси.

Для написання додатку для платформи Android було обрано Java, так як це основна мова для розробки додатків для Android[10]. Це зумовлено тим що сама операційна система реалізована віртуальній машині Java (JVM). Java це кросплатформенна об’єктно-орієнтована мова програмування. Особливістю

цієї мови є використання JVM, що дозволяє запускати код програми майже на будь-якому пристрої.

В ролі середовища розробки обрано Android Studio, так як це основний інструмент для розробки додатків. Android Studio надає ряд зручних та корисних утиліт, таких як встроєний емулятор Android, можливість графічного редактуваннння макету (layout) додатків, велику бібліотек готових різних шаблонів типових рішень дизайну та інші інструменти. На рис. 2.16. показано інтерфейс проекту у Android Studio.

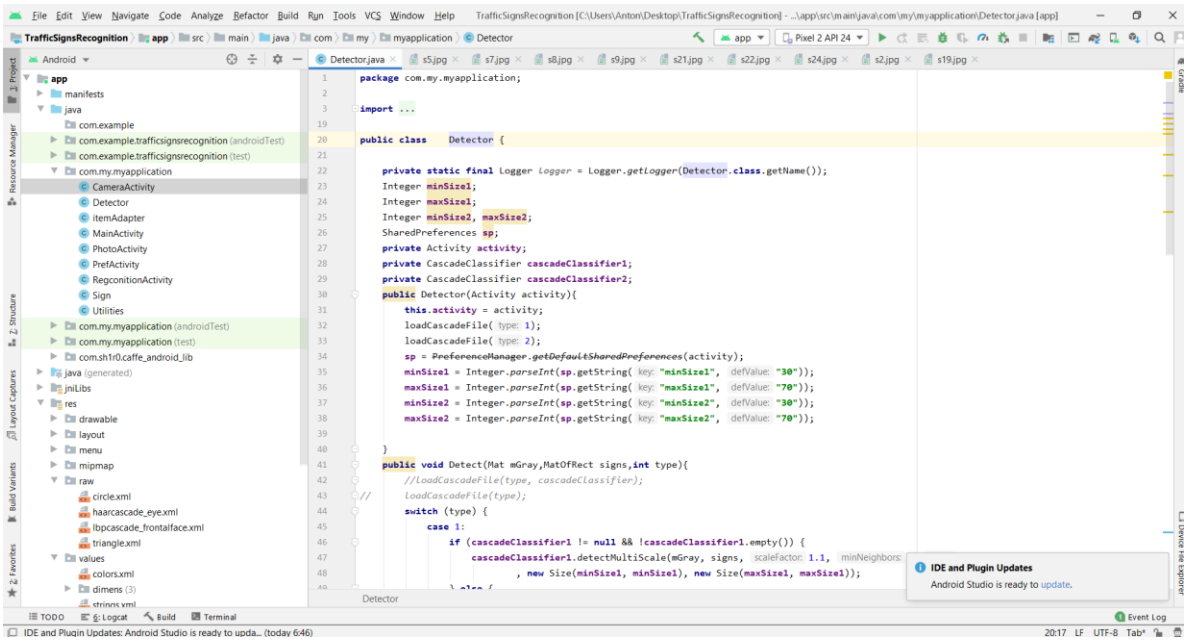


Рис 2.16. Вікно середовища розробки Android Studio

2.4.2. Вибір засобів для роботи з зображеннями.

MATLAB – пропрієтарний мультиплатформовий пакет прикладних програм розроблений для вирішення задач технічних та наукових обчислень. Мова MATLAB є високорівневою інтерпретованою мовою програмування. Для роботи з зображеннями надається пакет Image Processing Toolbox.

Даний пакет надає великий ряд інструментів для роботи з зображеннями включаючи як базові операції – поворот зображення, бінаризацію, методи покращення зображення та інші, так і продвинуті – різноманітні алгоритми згладжування, екстраполяція зображень.

MATLAB підтримує векторні і растрові типи зображення, а програмно вони представлені типами double і uint8. Векторні зображення описуються

наборами графічних примітивів, а растрові в MATLAB діляться на бінарні, напівтонові, палітрові і повнокольорові.

Елементи бінарних зображеннь приймають 0 або 1, в основному вони отримуються шляхом обробки інших типів.

Напівтонові зображення складаються з елементів, які можуть приймати одне із значень інтенсивностей якогось кольору, в більшості випадків використовується глибина кольору в 8 біт.

В палітрових зображеннях значення пікселей є посиланням на комірку карти кольорів (палітру).Палітра представляє собою двовимірний масив, в стовбцях якого знаходяться інтенсивності кольорових компонент одного кольору.

Табл. 2.1. Огляд типів в MATLAB

Тип зображення	double	Uint8
бінарне	0 і 1	0 і 1
полутонове	[0,1]	[0,255]
палітрове	[1,розмір палітри] Де 1 – перший рядок палітри	[0,розмір палітри] Де 0 – перший рядок палітри
повнокольорове	[0,1]	[0,255]

Також у MATLAB реалізовано каскадні детектори, в тому числі і метод Віола-Джонса системним об'єктом `vision.CascadeObjectDetector`. Даний клас надає методи для використання детекторів та надає можливість їх тренувати. Крім того даний клас містить претреновані каскади, наприклад детектори для розпізнавання обличчя та його рис.

Отже, MATLAB є доволі хорошим інструментом для роботи з зображенням, проте даний пакет інструментів має ряд мінусів, а саме.

- MATLAB є інтерпретованою мовою програмування, що зумовлює низьку швидкодію коду. Окрім цього, більшість команд не підтримують виконання їх на графічному процесорі. Вкупі ці проблеми роблять

малопридатним цей пакет в цілях CV , де є важливим робота в реальному часі. Виклик функцій MATLABу у Python теж є повільним і громіздким, для виклику потрібно встановити спеціальні бібліотеки.

- MATLAB є доволі дорогим, крім цього більшість алгоритмів є закритими, що означає, що користувач не може бачити код більшості алгоритмів і користувач має вірити, що алгоритм реалізовано правильно.
- Погана переносимість коду – скомпільований користувачем код коректно запускається лише на ідентичних версіях MATLABу.

OpenCV (Open Source Computer Vision Library)– одна із найбільших бібліотек алгоритмів комп'ютерного зору, обробки зображень і чисельних методів з відкритим кодом. OpenCV була створена для забезпечення загальної інфраструктури додатків заснованих на комп'ютерному зорі для спрощення розробки і вирішення більшості типових задач. Бібліотека випущена під BSD-ліцензією, будь-який бажаючий може її використати для комерційної розробки або в свої цілях. Поряд з відомими компаніями, такими як Google, Yahoo, Microsoft,Intel,IBM та іншими компаніям,бібліотека популярна серед стартапів так .

Бібліотека містить понад 2500 оптимізованих алгоритмів для обробки і аналізу зображення і відео, такі як розпізнавання і виявлення об'єктів, відстежування руху. Сирцевий код бібліотеки написаний на C++, проте бібліотека підтримується багатьма мовами такими, як Java, Python, Ruby, C# та інші.

Бібліотека складається з багатьох модулів, найважливішими з яких є:

- Opencv_core – ядро бібліотеки, містить базові стуктури, базові алгоритми (генерація псевдовипадкових чисел, різні види перетвореннь Фур'є та інші)
- Opencv_imgproc – обробка зображеннь (фільтри, перетворення та інші)
- Opencv_video – аналіз відео і відслідковування об'єктів (оптичний потік, шаблони руху)
- Opencv_highgui – простий UI

- Opencv_ml – методи і різні моделі машинного навчання (SVM, дерева прийняття рішень)
- Opencv_objdetect – детектування образів на зображенні (ознаки Хаара, HOG)
- Opencv_gpu – прискорення функцій opencv за рахунок CUDA.
- Крім того підтримуються бібліотеки розпаралелювання – OpenMP і TBB.

Метод Віола-Джонса реалізований в OpenCV за допомогою інструменту opencv_traincascade і виконується прямо з командного рядка. Даний інструмент знаходиться в директорії бібліотеки і для його використання потрібно дві вибірки – позитивна, у якій для кожного зображення указано ROI(region of interest – область на якій є шуканий об'єкт) і негативна – різноманітні зображення, які не містять шукані об'єкти. Після створення цих вибірок, треба лише викликати утиліту у командному рядку, вказати шлях до вибірок та указати додаткові параметри такі, як:

- Кількість ступенів каскаду
- Розміри буферів пам'яті
- Ознаки – підтримуютья ознаки Хаара і LBP
- Для ознак Хаара можна вибрати різні набори примітивів.
- Розмір ознак, у пікселях
- Тип бустінгу – підтримується DAB (discrete AdaBoost), RAB (Real AdaBoost), LB (logitBoost), GAB (GentleBoost).

Отже, у програмі буде використано бібліотеку OpenCV, так як дана бібліотека надає всі необхідні інструменти для роботи з відео, зображенням та тренування каскадів. Крім цього бібліотека має інтерфейс для Python і Java.

2.4.3. Вибір програмних засобів для класифікації знаку.

Так як для написання нейромережі було вибрано Python, тому в даному розділі буде розглянуто лише фреймворки які підтримують цю мову.

TensorFlow - комплексна платформа для машинного навчання с відкритим сирцевим кодом. Фреймворк був розроблений командою GoogleBrain і написаний на C і C++ з підтримкою CUDA. Особливістю TensorFlow є використання графів потоків даних, у яких вузли графів це математичні операції, а ребра графів це масиви даних (тензори). Крім того TensorFlow містить інструментарій візуалізації даних – TensorBoard.

Плюси цього фреймворку:

- Підтримка CUDA забезпечує високу продуктивність.
- Фреймворк бере на себе оптимізацію ресурсів для обчислень
- Так як TensorFlow є доволі популярним, більшість проблем з яким стикається користувач скоріш за все була вже вирішена

До мінусів можна віднести:

- Тяжкий в освоєнні, більша частина написаного коду буде шаблонним
- Погана документація
- Велика кількість підтримуваних версій, що заплутує користувача
- Необхідність контролювати відеопам’ять

Microsoft CNTK - це швидкий, надійний та універсальний з відкритим сирцевим кодом. Особливістю цього фреймворка є оптимізація коду та порівняно менші вимоги до ресурсів пам’яті, що робить його ефективним інструментом для швидкого навчання та розробки нейронних мереж, що в купі з інтеграцією масивів даних Microsoft,робить його одним із найкращих в своєму роді.

Плюси цього фреймворку:

- Один із найшвидших і найточніших фреймворків
- Хороша масштабованість
- Підтримка найновіших технологій в області машинного навчання

Мінуси:

- Відсутність засобів візуалізації
- Відсутня підтримка процесів ARM

PyTorch – середовище машинного навчання з відкритим сирцевим кодом, забезпечує тензорні обчислення з використанням GPU. Особливістю фреймворка (рис. 2.17.) є динамічний граф обчислень, тобто є можливість перебудовувати граф обчислень під час самих обчислень

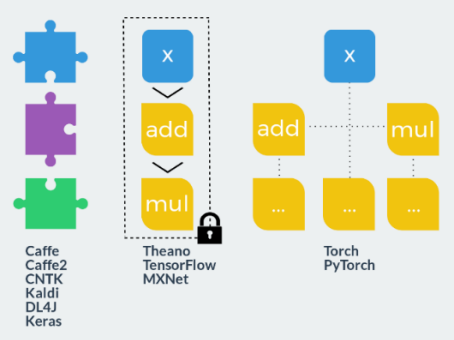


Рис 2.17.Порівняння структури обчислень PyTorch з іншими фреймворками.

Плюси:

- Має широкий вибір готових моделей
- Простота модифікації встроєних типів, наприклад є можливість створювати свої типи шарів
- Швидкодія

Мінуси:

- Неповна документація
- Відсутність засобів візуалізації
- Недостатня підтримка моделей

Caffe (Convolutional Architecture for Fast Feature Embedding) – фреймворк розроблений Facebook, написаний на C++ і з інтерфейсом на Python. Фреймворк орієнтований на використанні в області розпізнавання та класифікації зображення.

Плюси:

- Швидкий, масштабуємий і займає мало місця.
- Має велику кількість готових рішень – Model zoo.

Мінуси:

- Мала швидкодія при роботі з великими і комплексними масивами даних
- Обмежена підтримка спільнотою

Keras – мінімалістична бібліотека машинного навчання, заснована на Python. Особливістю фреймворка є можливість запуску поверх TensorFlow, Theano та CNTK. Фреймворк націлений на оперативну роботу з нейромережами, є компактними, модульним, існує можливість використовувати моделі створені іншими фреймворками, наприклад Caffe.

Плюси:

- Простота і зручність використання
- Хороша і повна документація
- Інтегрований в TensorFlow
- Має вбудовану підтримку відеокарт від Intel та AMD і підтримує навчання на декількох графічних прискорювачах.

Мінуси:

- Немає можливості тонкого налаштування блоків нейромереж.
- Не підходить для масштабних проектів.

Отже, з усіх оглянутих фреймворків вирішено було вибрати Keras через свою простоту,швидкість розробки програми.

2.4.4. Аналіз даних для моделі

Для навчання моделі класифікатора і каскадного детектора було вибрано відкритий набір даних – GTSRB (German Traffic Sign Recognition Benchmark)[11], відкрита база німецьких дорожніх знаків. Було обраний цей датасет, так які більшість знаків практично не відрізняються від українських знаків. Окрім цього для тестування було обрано GTSDB (German Traffic Sign Detection Benchmark) – також набір німецьких знаків, але які не вирізані із зображення.

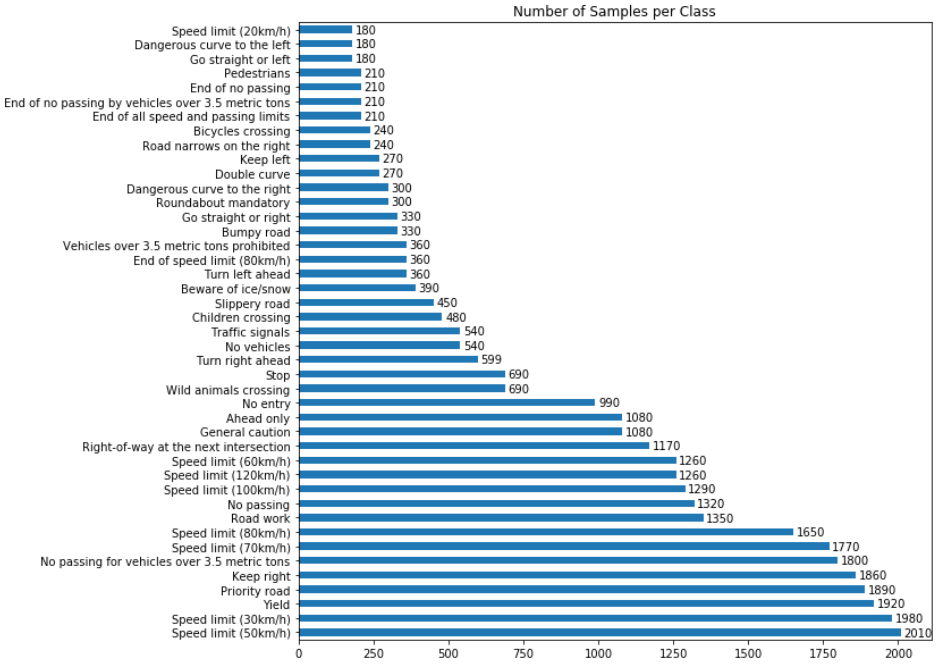


Рис.2.18. Розподіл зображень по класами.

GTSRB містить 43 класи зображень і більше ніж 50000 їх зображень. Розподіл зображень по класам показано на рис. 2.18.Приклад знаків з вибірки показано на рис. 2.20а. Для кожного із зображень є анотація у якій вказано шлях до зображення, розміри зображення, клас зображення та 2 пари координат – координати обмежувального прямокутника(рис.2.19.) у якому знаходиться знак на зображенні.

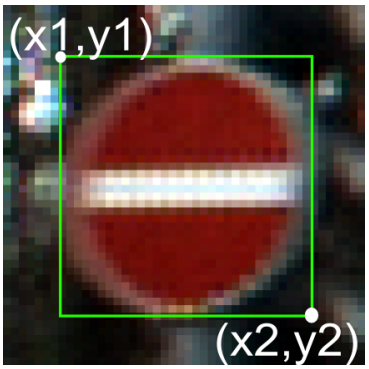


Рис.2.19.Приклад координат обмежувального прямокутника

А) Б)

Рис.2.20.Приклад зображень з а)позитивної вибірки і б)негативної вибірки.

Так, як для метода Віола-Джонса потрібна також вибірка негативних зображень, то в якості вибірки було вибрано різноманітні зображення(рис.2.20б.), що не містять знаків або подібних до них елементів. Так, як метод не використовує колір знаку, вибірка місить лише чорно-білі зображення.

ВИСНОВКИ ДО РОЗДІЛУ 2

Отже, в даному розділі було вибрано алгоритми для вирішення поставлених задача. Для виконання задачі локалізації було вибрано метод Віола-Джонса через те, що даний метод має готову реалізації в бібліотеці OpenCV, легко навчається і має достатню точність і швидкість роботи. Для виконання задачі класифікації було вибрано згорткову нейронну мережу, як один із найефективніших засобів для класифікації зображення.

В якості набору даних для навчання детектора і класифікатор було вибрано німецький набір зображень дорожніх знаків – GTSRB.

Для написання програм тренування нейронної мережі та різних скриптів було обрано мову Python і бібліотеку Keras . Для створення додатку для Android було обрано мову програмування Java і середовище розробки Android Studio. Для обробки зображень обрано бібліотеку OpenCV.

РОЗДІЛ 3.
РОЗРОБКА ТА ТЕСТУВАННЯ ДОДАТКУ

В даному розділі буде виконана розробка додатка та його складових, а саме систем локалізації та класифікації зображення. Розробку буде виконано в кілька етапів:

1. Розробка детектора (метод Віола-Джонса).
2. Розробка класифікатора (згорткова нейронна мережа).
3. Розробка додатку.
4. Тестування додатку.

3.1. Розробка системи локалізації

Так, як знаки можуть значно відрізнятися за формою,використання одного каскаду буде недоцільним, адже він буде мати низьку точність. Тому вирішено було натренувати два каскади – один для круглих знаків (забороняючі), інший для знаків трикутної (попереджувальні) та квадратної форми(“Пішохідний перехід”).

Для навчання даних каскадів було сформовано дві вибірки, позитивна та негативна. Позитивна вибірка була сформована так – за допомогою python-скрипту випадковим чином було вибрано 1500 зображень знаків з датасету GTRSB для кожного із каскадів. При цьому було кожне зображення знаку було трансформовано в градації сірого та змінено анотації – замість координатів обмежувального прямокутника, вказується координати верхньої лівої точки обмежувального прямокутника та висота і розмір знака на зображенні.

Для подальшого потрібно використати спеціальну утиліту OpenCV - *opencv_createsamples,*яка знаходиться в директорії бібліотеки. Дана утиліта надає можливості для генерації, запису та перегляду датасету. Для створення позитивної вибірки в командному рядку було викликано *opencv_createsamples.exe -vec vector.vec -bg good.dat -num 1500*, де:

- -vec vector.vec – вихідний файл
- -bg good.dat – файл у якому вказано шлях до зображень, кількість шуканих об’єктів (знаків) та координати об’єктів

- -num 1500 - кількість зображень

Після підготовки вхідних даних, викликається утиліта для тренування каскаду, яка теж знаходиться в директорії OpenCV – *opencv_traincascade.exe*. Аналогічно до попередньої утиліти, ця викликається в командному рядку. Для тренування каскаду програму було викликано з такими параметрами:

- -data D:\GTSRB\round_casc – шлях до папки, у якій буде збережено каскад
- -vec vector.vec – шлях до файлу з позитивною вибіркою
- -bg negatives.dat – шлях до файлу з негативною вибіркою
- -numStages 16 – максимальна кількість ступенів каскаду
- -minHitRate 0.999 – значення, яка визначає якість навчання, по суті (1-0.999) =0.1%– кількість пропущених об’єктів на ступені
- -maxFalseAlarmRate 0.4 – рівень помилкової тривоги, при заданому досягнені заданого рівня тривоги, навчання зупиняється
- -numPos 1244 – кількість позитивних зображень які використовується на кожному рівні навчання, вказано менше реальної вибірки, так як на кожному рівні кількість зображень збільшується.
- -numNeg 3000 – кількість негативних зображень.
- -precalcValBufSize і -precalcIdBufSize 2048 – розмір буферів під пам'ять в МБ
- -acceptanceRatioBreakValue 10e-5 – значення, яке унеможливлює перенавчання каскаду, при його досягненні, навчання зупиняється
- -featureType - ознаки, які використовуються, LBP або ознаки Хаара
- -bt - тип бустінга, GentleAdaBoost, RealAdaBoost або DiscreteAdaBoost

Для вибору найкращих параметрів було натреновано кілька різновидів каскадів з різними параметрами, які наведено в таблиці 3.1. та таблиці 3.2.

Каскади навчалися на машині з такими параметрами:

- Intel Core i7-9750H CPU 2.6 ГГц, 6 ядер, 12 потоків
- 8 Гб ОЗУ

- GeForce GTX 1650 4 Гб відеопам’яті.

Табл.3.1.Порівняння натренованих каскадів для круглих знаків

№ каскаду	Ознаки	Точність	Кількість ступенів	Тип бустінга	Час навчання
1	Хаара, базові	69%	12	GentleAdaBoost	48 хв 45 с
2	Хаара, розширені	74%	18	GentleAdaBoost	3 год 25 хв
3	LBP	71%	12	GentleAdaBoost	1 хв 23 с
4	LBP	65%	14	LogitBoost	2 хв 51с
5	LBP	61%	11	RealAdaBoost	1 хв 9 с
6	LBP	60%	11	DiscreteAdaBoost	1 хв 16 с

Табл.3.2.Порівняння натренованих каскадів для трикутних знаків

№ каскаду	Ознаки	Точність	Кількість ступенів	Тип бустінга	Час навчання
7	Хаара, базові	74%	13	GentleAdaBoost	59 хв 15 с
8	Хаара, розширені	80%	18	GentleAdaBoost	3 год 10 хв
9	LBP	82%	19	GentleAdaBoost	10 хв 23 с
10	LBP	75%	15	LogitBoost	5 хв 30с
11	LBP	70%	13	RealAdaBoost	2 хв 19 с
12	LBP	73%	14	DiscreteAdaBoost	2 хв 40 с

Виходячи з отриманих даних, було вирішено використовувати в додатку другий та дев’ятий каскад, так як вони показали найкращі результати. Отриманої точності достатньо, адже якщо в поточному кадрі знака не буде знайдено, його можна буде знайти в наступному. Важливіше було зменшити

кількість помилкових об’єктів а ніж не знайдених знаків, так як це часта локалізація помилкових об’єктів призводить до подальшої їх класифікації, що зменшує швидкодію додатка. Нижче на рис.3.1, рис.3.2. і рис.3.3. наведено приклад роботи детектора та типові помилки локалізації.

Рис.3.1.Знак є на зображенні, але його не знайдено.

Рис.3.2.Знаки успішно знайдено на зображенні.

Рис.3.3.Посторонні об’єкти локалізовано як знаки.

Було помічено, що більшість помилок пов’язано з малим розміром знака на зображенні (менше 30 на 30 пікселів), поворот знака під великим кутом, зашумленість або часткова видимість знака та сторонні об’єкти схожі за формою до шуканих часто локалізувались.

3.2. Розробка системи класифікації

Для навчання нейронної мережі була використано набір зображень GTRSB, про який було написано вище, всього навчальна вибірка містить 39209 зображень, а тестова – 12630 зображень. Перед процесом навчання зображені були преоброблені:

- Знаки було вирізано по обмежуючому контуру .
- Так, як знаки в навчальній вибірці мають розмір від 15 х 15 до 200 х 200 на пікселів, потрібно було привести їх до одного розміру. Експериментальним шляхом було обрано 30 х 30 і 32 х 32 пікселя та натреновано дві нейромережі, архітектури яких вказано у таблиці 3.3. та таблиці 3.4. Знаки було масштабовано за допомогою бікубічної інтерполяції.
- Отримані зображення перетворено в градації сірого кольору, так як це прискорює роботу нейронної мережі – замість трьох карт для кожного кольорового каналу використовується лише один.
- Для зниження впливу коливання контрасту на зображені, було використано контрастне вирівнювання CLAHE (Contrastlimited adaptive histogram equalization) [12]. Цей метод покращує контраст напівтонового зображення шляхом перетворенням значень пікселей методом контрастно обмеженої адаптивної еквалазації гістограми. Контраст кожної частини зображення підвищується, що пов’язано зі зміною форми гістограми. Приклад використання CLAHE вирівнювання зображено на рис.3.4.

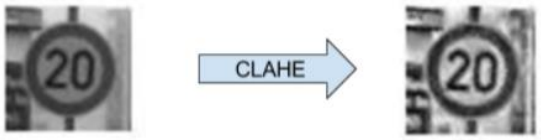


Рис. 3.4.Результат CLAHE вирівнювання

Розмір вхідних даних і архітектуру нейронної мережі було обрано експериментальним шляхом. Було вирішено створити кілька різних нейроннних мережі і обрати найкращу з них.

Табл 3.3.Архітектура CNN1:

Шар	Тип	Кількість карт та нейронів
0	Вхідний	1 карта 32 х 32 нейрона
1	Згортковий	12 карт 28 х 28 нейронів з вікном 5 х 5
2	Субдискретизації	12 карт 14 х 14 нейронів з вікном 2 х 2
3	Згортковий	24 карти 10 х 10 нейронів з вікном 5 х 5
4	Субдискретизації	24 карти 5 х 5 нейронів з вікном 2 х 2
5	Повнозв’язний	150 нейронів
6	Dropout(0.5)	
7	Повнозв’язний	100 нейронів
8	Dropout(0.5)	
9	Повнозв’язний	26 нейронів
10	softmax	

Табл 3.4.Архітектура CNN2:

Шар	Тип	Кількість карт та нейронів
0	Вхідний	1 карта 30 х 30 нейрона
1	Згортковий	32 карти 26 х 26 нейронів з вікном 5 х 5
2	Згортковий	32 карт 22 х 22 нейронів з вікном 5 х 5
3	Субдискретизації	32 карти 11 х 11 нейронів з вікном 2 х 2
4	Dropout(0.5)	
5	Згортковий	64 карти 9 х 9 нейронів з вікном 3 х 3

Продовження Табл 3.4.Архітектура CNN2:

Шар	Тип	Кількість карт та нейронів
6	Згортковий	64 карти 7 x 7 нейронів з вікном 3 x 3
7	Субдискретизації	64 карти 3 x 3 нейронів з вікном 2 x 2
8	Flatten	576 нейронів
9	Повнозв’язний	256 нейронів
10	Dropout(0.5)	
11	Повнозв’язний	26 нейронів
12	softmax	

У обох нейронних мережах використовувалась чередування згорткових шарів та шарів субдискретизації, на виході отримувалися повнозв’язні шари. На всіх шарах в ролі функції активації використовувалась ReLU.

Також використовувалися такі шари як:

- Dropout(0.5) – спеціальний шар, у якому з ймовірністю 0.5 будь-який нейрон може “відмерти”. Даний шар використовується для того, зменшити можливість перенавчання нейронів.
- Flatten – шар, який приводить дані до одновимірної розмірності
- На виході використовувався шар softmax, для того щоб отримати дані для інтерпретації. Значення шару знаходяться в інтервалі від 0 до 1, а значення нейронів визначаються за формулою:

$$y_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}} \qquad (3.1.)$$

де y_i – значення нейрона на виході, z_i – значення нейрона на попередньому шарі.

Обидві нейронні мережі навчалися на тій же самій машині, що у попередньому розділі. Навчання тривало по 35 епох і за результатами навчання кращі показники має CNN1. Хоч вона має і меншу точність , проте

має кращу швидкодію, що є критично важливим у вирішенні поставленої задачі.

Табл 3.5.Порівняння результатів роботи отриманих нейромереж.

Показники	CNN1	CNN2
Точність	95.82%	97.35%
Час класифікації	0.8 мс	2.1мс

Створена нейронна мережа розпізнає нижче наведені знаки:

- 8 знаків обмеження максимальної швидкості
- “Пішохідний перехід”
- Попереджувальні знаки - “Світлофорне регулювання”, “Аварійно небезпечна ділянка”, “Небезпечний поворот ліворуч/праворуч”, “Декілька поворотів”Слизька дорога”,”Нерівна дорога”, “Звуження дороги”,”Дорожні роботи”.
- Заборонні знаки - “Рух заборонено”, ”Рух вантажних автомобілів заборонено””, “Зупинку заборонено”, “Стоянку заборонено”,”Обгін вантажним автомобілями заборонено”, ”Обгін автомобілями заборонено”, “Стоп”, “В’їзд заборонено”.
- Всього 26 знаків.

3.3. Розробка додатку

3.3.1. Реалізація алгоритму роботи системи.

Створення додатку було розпочато з побудови алгоритму обробки відео. Для використання натренованих каскадів та нейромережі використано засоби бібліотеки OpenCV. Для каскадів викликається функція *detectMultiScale* з такими параметрами:

- image – зображення на якому буде виконуватись пошук знаків
- scaleFactor – коефіцієнт збільшення скануючого вікна, даний параметр відповідає за крок збільшення вікна. Експериментально було обрано значення 1.1 як оптимальне між швидкістю та точністю.

- minNeighbors – кількість сусідів, використовується для того щоб каскад не видавав в якості відповіді сусідні області, які знаходяться між собою на відстані декількох пікселів.
- minSize – мінімальний розмір вікна, обрано за замовчування 30 x 30
- maxSize – максимальний розмір вікна, обрано за замовчуванням 70 x 70

На виході отримуємо ROI шуканого знака.

Для пришвидшення локалізації зображення вирішено було обробляти за замовчуванням праву верхню частину зображення, як показано на рис 3.5. Це зумовлено тим, що зазвичай знаки розташовані на правій стороні дороги і знаходяться достатньо високо.

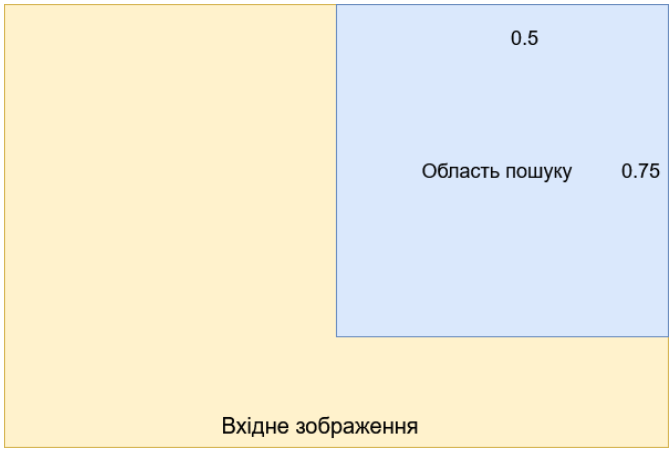


Рис 3.5.Зона пошуку на зображенні

Використання такого підходу пришвидшує етап локалізації до 2 разів. Також було вирішено проводити локалізацію таким чином – парні кадри оброблюються першим каскадом (попереджувальні знаки), а непарні оброблюються другим каскадом (заборонні знаки).

Для роботи з нейромережами було використано модуль dnn OpenCV. Так як OpenCV не підтримує формат “.h5” моделі Keras, нейромережу було сконвертовану до формату TensorFlow. Для завантаження моделі використовується метод *readNetFromTensorflow*, який приймає два параметри – файл з структурою моделі та файл з вагами кожного нейрона моделі.

Отже, алгоритм обробки кадру можна представити таким чином:

1. Отримуємо кадр з камери
2. Вирізаємо область в якій проводимо пошук

3. Локалізуємо знаки детектором відповідно до номера кадру
4. Отримуємо ROI знаків, виводимо на екран їх границі
5. Вирізаємо знайдені знаки
6. Трансформуємо колір отриманих вирізаних зображень в градації сірого
7. Використовуємо CLAHE
8. Змінюємо розмір області на 32 x 32 пікселі
9. Використовуємо нейронну мережу
10. Виводимо клас знаку і ймовірність цього знаку на екран

3.3.2. Реалізація Android додатку

Для показу можливостей реалізованої системи було створено додаток, який надає базові можливості такі, як використання системи в реальному часі чи можливість обрати або зробити фото для подальшої обробки. Для початку роботи додатку потрібно надати йому дозволи для використання камери та сховища. Стартове меню показано на рис. 3.6.

Рис 3.6. Головне меню додатку

Для того щоб розпочати роботу в реальному часі потрібно натиснути на ‘Run in real-time’. У відкритому вікні передається зображення з камери, на якій відрисовуються, у разі знаходження, границі знаків, а у колонці справа розміщуються зображення знаків, які знаходяться в області пошуку на даний момент. Інтерфейс камери показано на рис. 3.7.

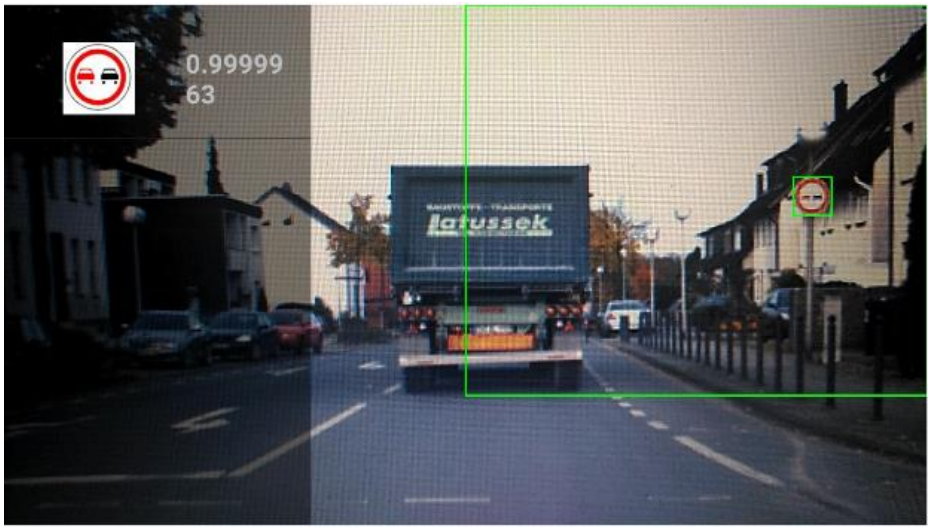


Рис 3.7.Приклад роботи додатку

Для вибору фото потрібно в головному меню обрати ‘Pick photo’ яке надає можливості для вибору з сховища телефону чи зробити фото з камери. Інтерфейс меню показано на рис.3.8.

При цьому результатом аналогічне вікно з відрисованими знаками та їх границями.

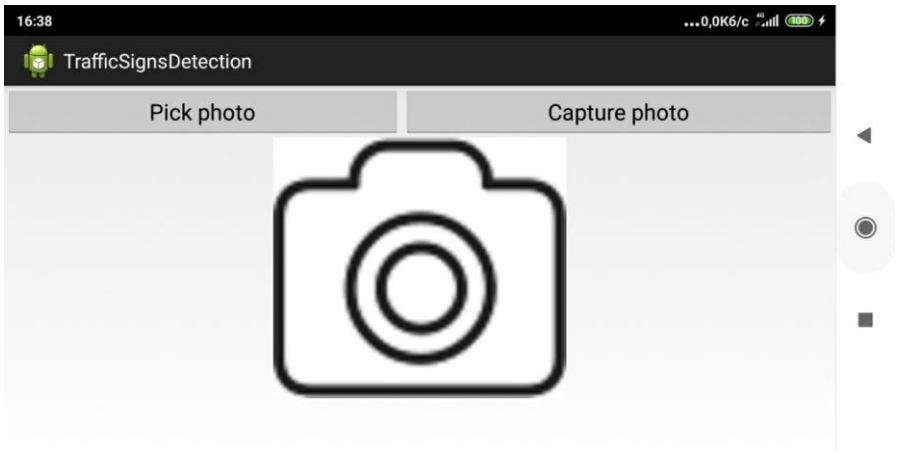


Рис.3.8. Меню ‘Pick photo’

Також реалізована настройка таких параметрів як розміри пошукового вікна у детекторі та можливість зміни області пошуку. Для доступу в меню настройок(рис.3.9.) потрібно натиснути в праву верхню частину.

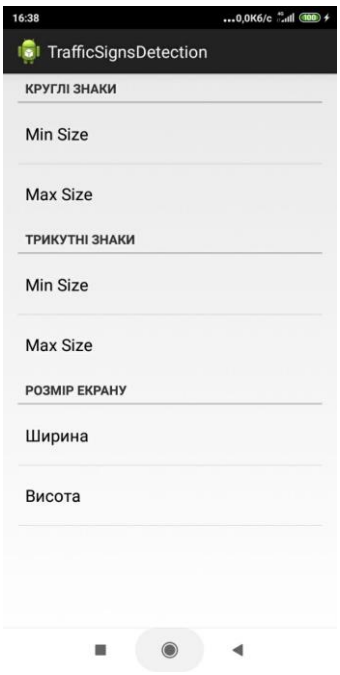


Рис. 3.9. Меню настройок

3.4. Опис методів та функцій головних класів додатку

Головним класом програми є клас *MainActivity.class* – саме через цей клас запускається сам додаток. Даний клас містить такі методи:

- public void Initialize() – метод, який ініціалізує кнопки ‘Pick photo’ та ‘Run in real-time’
- public void onCreate() – перевизначає метод onCreate, викикається при запуску додатка та створює кнопки кнопки ‘Pick photo’ та ‘Run in real-time’
- public void onClick() – реалізує логіку натискання на кнопки Pick photo’ та ‘Run in real-time’

Для роботи з каскадами використовується клас *Detector.class.* Даний клас містить такі методи:

- private void loadCascadeFile(int type) – метод, який завантажує каскади з .xml файлів, на вхід приймає число 1 або 2, яке відповідає за завантаження відповідного каскаду – для круглих або трикутних знаків.
- public void Detect(Mat mGray,MatOfRect signs,int type) – метод, який локалізує знак на зображенні, 1 параметр - зображення, 2 параметр – вектор, у якому буде зберігатись результат, 3 параметр – номер каскаду

За роботу з камерою та трансформацію зображення відповідає клас *CameraActivity.class.* Даний клас містить такі методи:

- private void Initialze() – метод, який ініціалізує параметри камери
- protected void onCreate() - перевизначає метод onCreate, викикається при запуску додатка.
- public void onResume() - перевизначає метод onResume, завантажує бібліотеку OpenCV.
- public Mat onCameraFrame(CvCameraViewFrame inputFrame) – локалізує зображення на вхідному кадрі і відрисовує обмежуваючий контур на екрані.

- public void Draw(Rect[] facesArray) – метод, який виконує етап класифікації, на вхід приймає вектор з ROI локалізованого знака.

3.5. Тестування додатку

Тестування додатку проводилося на двох мобільних телефонах – Xiaomi Redmi 5 та Zebra TC 56. Було підраховано затрачений час на етапах локалізації та класифікації знаку, підраховано локалізовані та не локалізовані знаки, локалізовані хибні об’єкти, правильно і неправильно класифіковані знаки. Загалом потрачений час на етапи локалізації та класифікації був таким:

- Використання всього зображення – 200-250 мс на Xiaomi Redmi 5 і 160 - 180 мс на Zebra TC 56.
- Використання запропонованої частини зображення – 40 -60 мс і 30 -50 мс відповідно
- Класифікація одного знаку на обох пристроях зайняла не більше 10 мс.

Загалом обробка одного кадру становила 60 – 110 мс на Xiaomi Redmi 5 і 50 -80 мс Zebra TC 56, що еквівалентно 9-17 кадрам в секунду і 13-20 кадрам в секунду. Варто зауважити, що кадри в секунду просідали до малих значень лише при великій кількості знаків – більше 5-7 знаків в кадрі.

Для підрахунку точності роботи додатку було використано такі формули:

$$T_{\text{локалізації}} = \frac{N_{\text{локалізовних знаків}}}{N_{\text{всі знаки}}} \quad (3.2.)$$

$$T_{\text{класифікації}} = \frac{N_{\text{класифікованих знаків}}}{N_{\text{локалізовних знаків}}} \quad (3.3.)$$

де $T_{\text{локалізації}}$ – точність локалізації, відношення локалізованих знаків до всіх знаків, а $T_{\text{класифікації}}$ – точність класифікації, відношення правильно класифікованих знаків до локалізованих знаків.

Тестування проводилося в світлий час дня на вулицях Києва і на обох вище названих пристроях. Всього на маршруті знаходилося 50 знаків і розглядалися лише знаки, що є в базі. В результаті тестування було отримано такі результати:

Табл. 3.6. Результати тестування

Телефон	$T_{\text{локалізації}}$	Не локалізовано	Хибно локалізовано	$T_{\text{класифікації}}$	Неправильно класифіковано
Xiaomi Redmi 5	43 (86%)	7	4	32(74.4%)	11
Zebra TC 56	45(90%)	5	3	36(80%)	9

ВИСНОВКИ ДО РОЗДІЛУ 3

Отже, в даному розділі було виконано розробку та тестування додатку.

Виконано реалізацію обраного алгоритму локалізації зображення. Точність отриманих каскадів на тестовій вибірці становить 74% та 82%, що є задовільним для роботи додатку.

Виконано розробку і тренування згорткової нейронної мережі, точність якої на тестовій вибірці становить 95.82%

Було виконано розробку Android-додатка на мові Java. Протестовано роботу додатку на двох пристроях та отримано дані точності локалізації та класифікації. Точність локалізації 86%-90%, а точність класифікації 74%-80%. Розбіжності в точності з тестовою вибіркою можна пояснити такими факторами: у використаній тестовій вибірці знаки зазвичай мають доволі малий розмір, тому їх локалізація утруднена. Під реального тестування, відстань до знаку не була постійною, що підвищило точність;Різке падіння точності класифікації пов’язане з камерами мобільних телефонів. Сучасні камери хоч і мають велику роздільну здатність, але у них є проблеми зі зйомкою в русі, автофокусом та програмним забезпеченням. Із-за цього зображення з камери передається часто шумами та розпливчастим, що зменшує точність класифікації; Використання вибірки з німецькими знаками теж зменшило точність класифікації. Хоч німецькі знаки доволі схожі з українськими, проте все ж мають певні розбіжності, що вплинуло на точність.

ВИСНОВКИ

Результатом бакалаврської роботи став додаток для розпізнавання дорожніх знаків. Додаток було розроблено опираючись на існуючі аналоги, їх плюси та мінуси. Отриманий продукт має простий та інтуїтивно зрозумілий інтерфейс, має достатньо швидкодію для розпізнавання дорожніх знаків в реальному часі.

Для вирішення задачі розпізнавання знаків було проведено дослідження різних алгоритмів, методів та способів вирішення цієї задачі. Для вирішення цієї проблеми було вирішено розбити задачу на дві підзадачі. Для кожної з отриманих підзадач було обґрунтовано вибір алгоритму та вибрано інструмент для реалізації підзадачі. Для задачі локалізації було обрано метод Віола-Джонса, а в ролі інструменту обрано бібліотеку комп'ютерного зору OpenCV. Було вирішено використати 2 детектора для різних за формою знаків, після цього було натреновано ряд різних детекторів і обрано найкращі із них. Для вирішення задачі класифікації розроблено згорткову нейронну мережу. Як інструмент розробки і тренування було обрано бібліотеку машинного навчання Keras. Було натреновано дві нейронні мережі й обрано оптимальнішу з них. Тестування на вибірці показало доволі високі результати точності. Для демонстрації роботи алгоритмів було розроблено простий додаток для Android на мові Java. Додаток надає можливості як і використання в реальному часі , так і використання фото з сховища телефона. Також було реалізовано можливість настройки роботи, є можливість зміни розмірів області роботи системи , а також можливість тонкої настройки алгоритмів. Було проведено тестування в реальних умовах , з використанням двох різних телефонів, яке показало трохи гірші результати аніж на тестовій вибірці. Це можна пояснити недосконалістю камер пристроїв та відмінностями українських дорожніх знаків від німецьких, які було використано в ролі тренувальних.

При певній доробці системи, наприклад вибір більш ефективних алгоритмів, можливе подальше її використання у системах допомоги водію або наприклад у додатках-асистентах, які окрім розпізнавання знаків можуть

розпізнавати як й інших учасників дорожнього руху так й надавати корисну інформацію – відеореєстрацію, прокладання маршруту та інші можливості. Також можна реалізувати даний додаток на певній високопродуктивній вбудованій системи задля подальшого її використовування як підсистему круїз-контролю , авто-пілоту чи тієї ж самої системи допомоги водію.

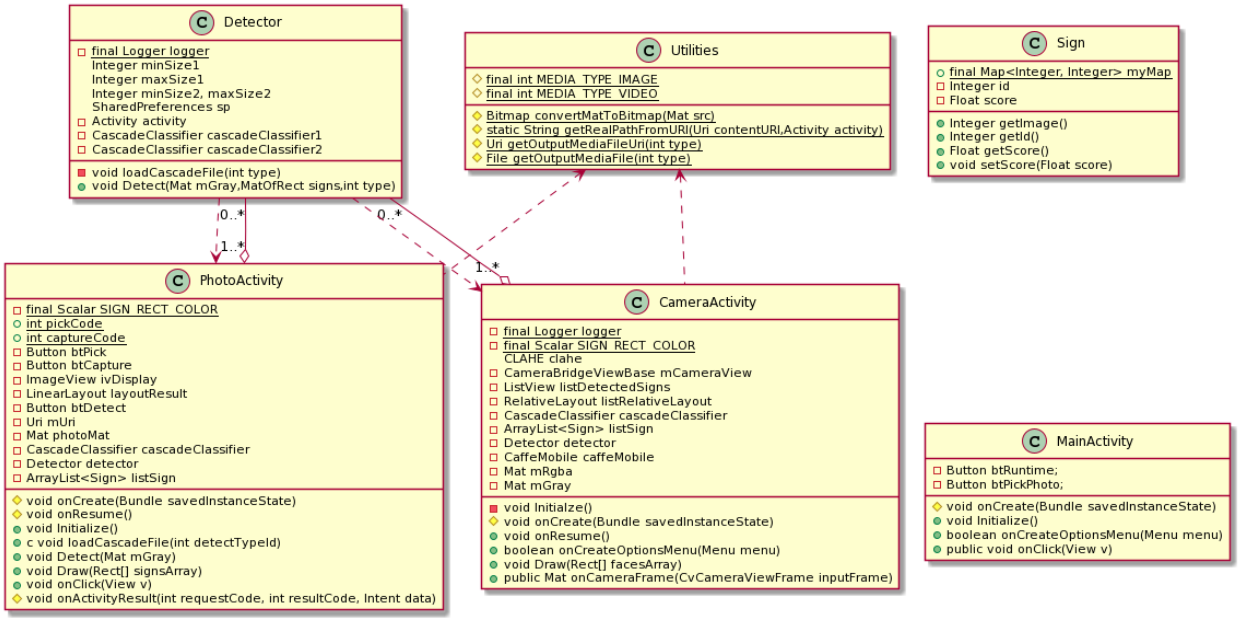
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Tesla solution for full self-driving [Електронний ресурс] – AnandTech, 2019 – Режим доступу: https://www.anandtech.com/show/14766/hot-chips-31-live-blogs-tesla-solution-for-full-self-driving
2. Rapid object detection using a boosted cascade of simple features / Paul Viola, Michael Jones // proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2001)– 2001 рік. – 11 с.
3. Применение локальных бинарных шаблонов к решению задачи распознавания лиц [Електронний ресурс] – Хабр, 2013 – Режим доступу: https://habr.com/ru/post/193658/
4. OpenCV шаг за шагом. Интегральное изображение [Електронний ресурс] - Robocraft, 2011 – Режим доступу: http://robocraft.ru/blog/computervision/536.html
5. Бустинг [Електронний ресурс] – MachineLearning.ru,2014 - Режим доступу: http://www.machinelearning.ru/wiki/index.php?title=%D0%91%D1%83%D1%81%D1%82%D0%B8%D0%BD%D0%B3
6. Алгоритм AdaBoost[Електронний ресурс] - MachineLearning.ru,2014 - Режим доступу: http://www.machinelearning.ru/wiki/index.php?title=AdaBoost
7. Глубокое обучение. Погружение в мир нейронных сетей / С. Николенко, Е. Архангельская, А. Кадурин - Питер, 2018// ISBN 978-5-4461-1537-2,480с.
8. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way[Електронний ресурс] – towards data science,2018 - Режим доступу: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
9. Обратное распространение ошибки[Електронний ресурс] - Университет ИТМО, 2019 - Режим доступу: neerc.ifmo.ru/wiki/index.php?title=Обратное_распространение_ошибки

10. Head First. Программирование для Android. 2-е изд. / Гриффитс Дэвид Марк, Гриффитс Дон – 2018 / ISBN: 978-5-4461-0708-7,912с/
11. German Traffic Sign Recognition Benchmark (GTSRB) [Електронний ресурс] -Institut fur neuroinformatik,2011- Режим доступу: http://benchmark.ini.rub.de/?section=home&subsection=news
12. CLAHE Histogram Eqalization – OpenCV [Електронний ресурс] - GeeksforGeeks, Режим доступу: https://www.geeksforgeeks.org/clahe-histogram-eqalization-opencv/

Додаток 1
до дипломного проєкту
на тему: «Додаток для розпізнавання дорожніх знаків»

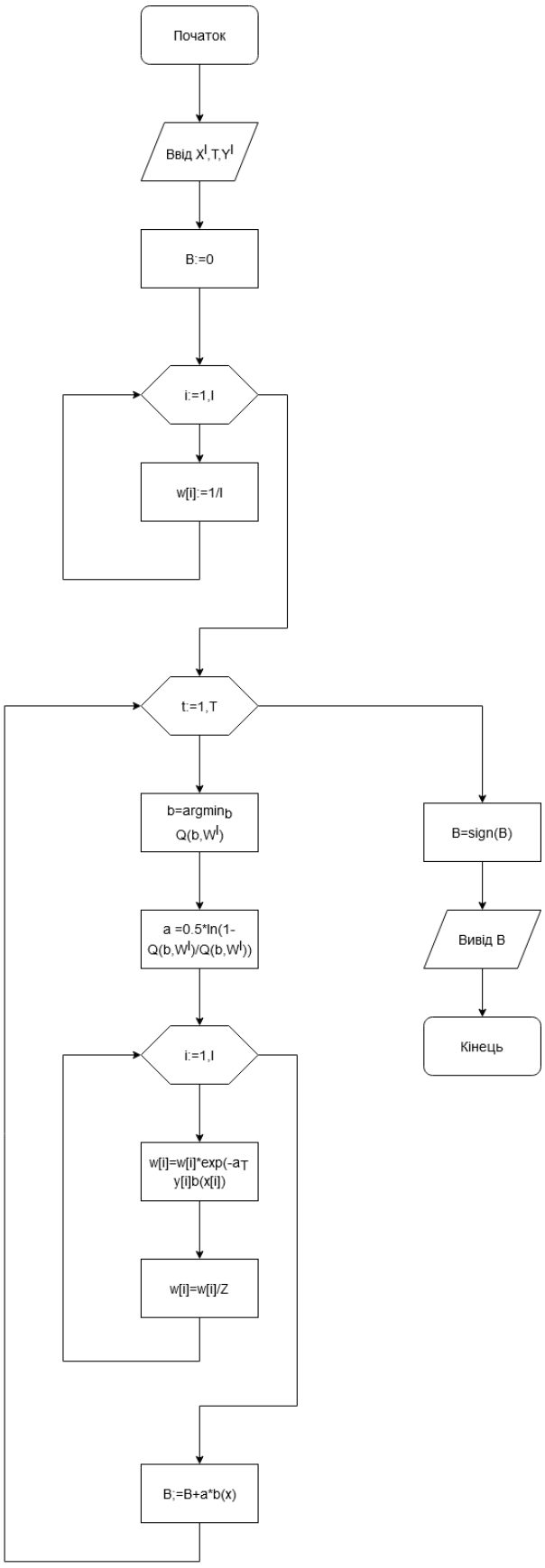
Київ – 2020 року



					ДП 4671. 03.000 Д1		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Бединський А. В.			Додаток для розпізнавання дорожніх знаків	Літ.	Аркуш 1 / Аркушів 1
Перевір.							
					Діаграма класів додатку	НТУУ “КПІ ім. Ігоря Сікорського”, ФІОТ, ІО-62	
Н. контр.		Сімоненко В.П.					
Затверд.							

Додаток 2
до дипломного проєкту
на тему: «Додаток для розпізнавання дорожніх знаків»

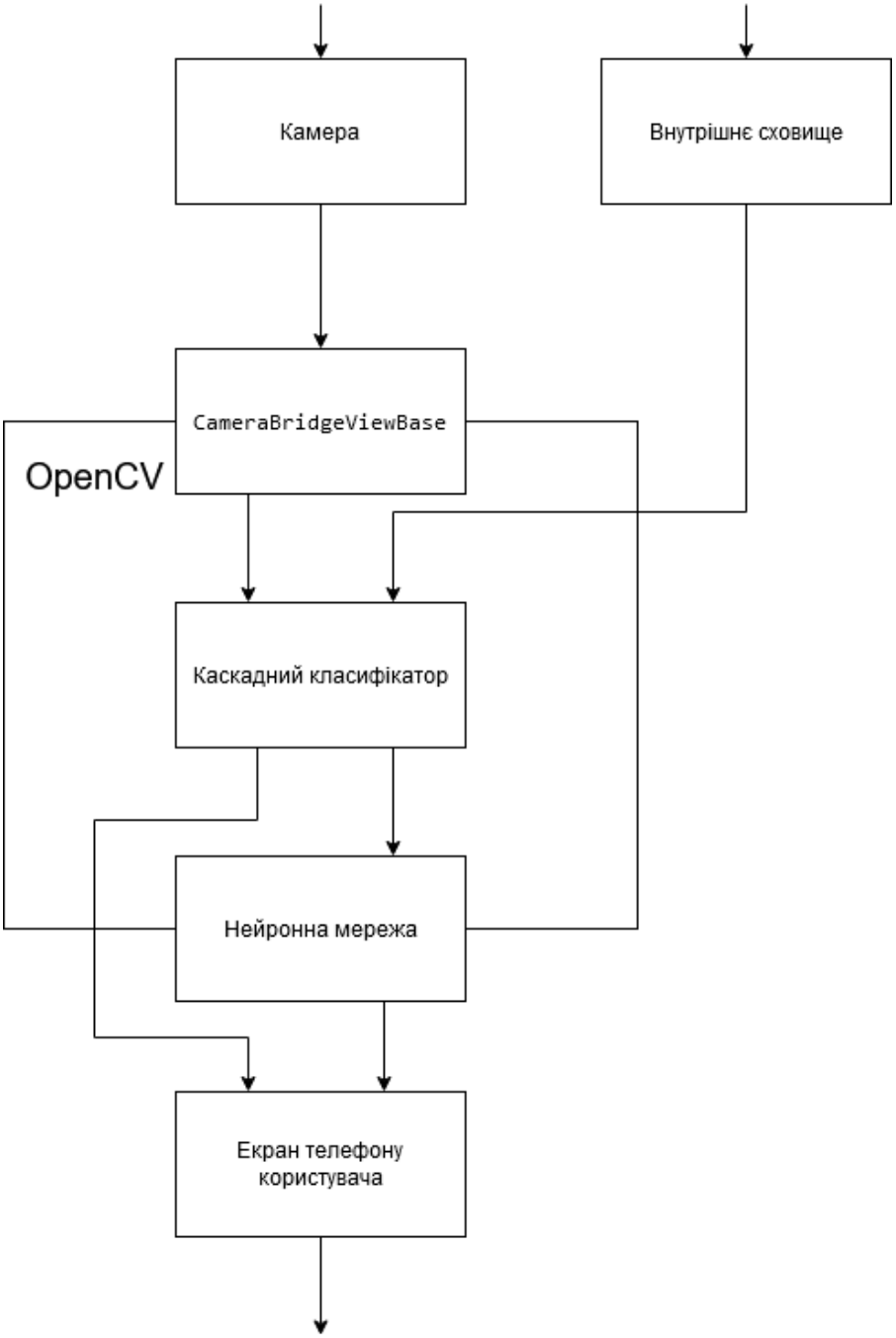
Київ – 2020 року



					ДП 4671. 03.000 Д2		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Бединський А. В.			Додаток для розпізнавання дорожніх знаків Метод Віола-Джонса	Літ.	Аркуш 1 / Аркушів 1
Перевір.							
						НТУУ “КПІ ім. Ігоря Сікорського”, ФІОТ, ІО-62	
Н. контр.		Сімоненко В.П.					
Затверд.							

Додаток З
до дипломного проєкту
на тему: «Додаток для розпізнавання дорожніх знаків»

Київ – 2020 року



					ДП 4671. 03.000 ДЗ			
Зм.	Арк.	№ докум.	Підпис	Дата				
Розробив		Бединський А. В.			Додаток для розпізнавання дорожніх знаків	Літ.	Аркуш	Аркушів
Перевір.							1	1
					Загальна структура системи	НТУУ “КПІ ім. Ігоря Сікорського”, ФІОТ, ІО-62		
Н. контр.		Сімоненко В.П.						
Затверд.								

Додаток 4
до дипломного проєкту
на тему: «Додаток для розпізнавання дорожніх знаків»

Текст програми

```
package com.my.myapplication;

import android.annotation.SuppressLint;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup.LayoutParams;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;

import org.opencv.android.BaseLoaderCallback;
import org.opencv.android.LoaderCallbackInterface;
import org.opencv.android.OpenCVLoader;
import org.opencv.core.Mat;
import org.opencv.core.MatOfRect;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;
import org.opencv.objdetect.CascadeClassifier;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;

//import org.opencv.highgui.Highgui;

public class PhotoActivity extends Activity implements OnClickListener {
   private static final Scalar FACE_RECT_COLOR = new Scalar(0, 255, 0, 255);
   public static int pickCode = 1;
   public static int captureCode = 2;
   private Button btPick;
   private Button btCapture;
   private ImageView ivDisplay;
   private LinearLayout layoutResult;
   private Button btDetect;
   private Uri mUri;
   private Mat photoMat;
   private CascadeClassifier cascadeClassifier;
   private Detector detector;
   private ArrayList<Sign> listSign;
   private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(this)
{
        @Override
        public void onManagerConnected(int status) {
```

```
            switch (status) {
                case LoaderCallbackInterface.SUCCESS:
                   photoMat = new Mat();
                   detector = new Detector(PhotoActivity.this);
                   break;
                default:
                    super.onManagerConnected(status);
                    break;
            }
        }
     };
    @Override
    protected void onCreate(Bundle savedInstanceState) {
       // TODO Auto-generated method stub
       super.onCreate(savedInstanceState);
       setContentView(R.layout.photo_layout);
       Initialize();
    }

    @Override
    protected void onResume() {
       // TODO Auto-generated method stub
       super.onResume();
       OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_1_0, this,
mLoaderCallback);
    }

    public void Initialize(){
       btPick = (Button)findViewById(R.id.btPick);
       btCapture = (Button)findViewById(R.id.btCapture);
       btDetect = (Button)findViewById(R.id.btDetect);
       ivDisplay = (ImageView)findViewById(R.id.ivDisplay);
       layoutResult = (LinearLayout)findViewById(R.id.layoutResult);
       //layoutResult.setVisibility(View.GONE);
       btDetect.setVisibility(View.GONE);
       btPick.setOnClickListener(this);
       btCapture.setOnClickListener(this);
       btDetect.setOnClickListener(this);
    }
    public void loadCascadeFile(int detectTypeId){
       try {
          InputStream is = null;
          File cascadeDir = getDir("cascade", Context.MODE_PRIVATE);
          File cascadeFile = null;

          switch (detectTypeId) {
          case 1:
             is =
getResources().openRawResource(R.raw.lbpcascade_frontalface);
             cascadeFile = new File(cascadeDir, "lbpcascade_frontalface.xml");
             break;
          case 2:
             is = getResources().openRawResource(R.raw.circle);
             cascadeFile = new File(cascadeDir, "traffic_signs.xml");
             break;
          case 3:
```

```
            is = getResources().openRawResource(R.raw.haarcascade_eye);
            cascadeFile = new File(cascadeDir, "haarcascade_eye.xml");
            break;
         default:
            break;
         }

         FileOutputStream os = new FileOutputStream(cascadeFile);
         byte[] buffer = new byte[4096];
            int bytesRead;
            while ((bytesRead = is.read(buffer)) != -1) {
                os.write(buffer, 0, bytesRead);
            }
            is.close();
            os.close();


            // Load the cascade classifier
            cascadeClassifier = new
CascadeClassifier(cascadeFile.getAbsolutePath());
      } catch (IOException e) {
         // TODO Auto-generated catch block
         e.printStackTrace();
      }
   }

   public void Detect(Mat mGray){

      Imgproc.equalizeHist(mGray, mGray);
      MatOfRect signs = new MatOfRect();
      listSign = new ArrayList<Sign>();

        detector.Detect(mGray, signs,1);
        Rect[] prohibitionArray = signs.toArray();
        Imgproc.cvtColor(photoMat, photoMat, Imgproc.COLOR_RGBA2BGR, 3);
        //Imgproc.cvtColor(photoMat, photoMat, Imgproc.COLOR_RGBA2BGR, 3);
        Draw(prohibitionArray);

        detector.Detect(mGray, signs,2);
        Rect[] dangerArray = signs.toArray();
       // Imgproc.cvtColor(photoMat, photoMat, Imgproc.COLOR_RGBA2BGR, 3);
       // Imgproc.cvtColor(photoMat, photoMat, Imgproc.COLOR_RGBA2BGR, 3);
        Draw(dangerArray);

        //get signs from photo
        ivDisplay.setImageBitmap(Utilities.convertMatToBitmap(photoMat));
    }
   public void Draw(Rect[] signsArray){

      for(int i = 0; i < signsArray.length; i++){
           Mat subMat = new Mat();
           subMat = photoMat.submat(signsArray[i]);

           ImageView ivv = new ImageView(this);
           ivv.setLayoutParams(new LayoutParams(LayoutParams.WRAP_CONTENT,
                    LayoutParams.WRAP_CONTENT));
```

```
            ivv.setImageBitmap(Utilities.convertMatToBitmap(subMat));

//          Sign.myMap.put("image"+i, Utilities.convertMatToBitmap(subMat));
//          Sign sign = new Sign("unknown", "image"+i);

//          listSign.add(sign);
            layoutResult.addView(ivv);
            btDetect.setVisibility(View.GONE);

//          layoutResult.setOnTouchListener(new OnTouchListener() {
//
//           @Override
//           public boolean onTouch(View arg0, MotionEvent arg1) {
//              // TODO Auto-generated method stub
//              Intent intent = new
Intent(PhotoActivity.this,RegconitionActivity.class);
//              intent.putParcelableArrayListExtra("key", (ArrayList<? extends
Parcelable>) listSign);
//              startActivity(intent);
//              return false;
//           }
//        });
          }
          //draw rectangle
          for (int i = 0; i < signsArray.length; i++){
           Imgproc.rectangle(photoMat, signsArray[i].tl(), signsArray[i].br(),
FACE_RECT_COLOR, 3);
          }


      }

      @SuppressLint("NewApi")
      @Override
      public void onClick(View v) {
         // TODO Auto-generated method stub
         switch (v.getId()) {
         case R.id.btPick:
            layoutResult.removeAllViews();
            Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
            intent.setType("image/*");
            startActivityForResult(intent, 1);
            break;
         case R.id.btCapture:
            layoutResult.removeAllViews();
            intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
            mUri = Utilities.getOutputMediaFileUri(Utilities.MEDIA_TYPE_IMAGE);
// create a file to save the video in specific folder (this works for video
only)
            intent.putExtra(MediaStore.EXTRA_OUTPUT, mUri);
            startActivityForResult(intent, captureCode);
            break;
         case R.id.btDetect:
            String photoPath =
Utilities.getRealPathFromURI(mUri,PhotoActivity.this);
            photoMat= Imgcodecs.imread(mUri.toString());
```

```
            Mat mGray = new Mat();
            Imgproc.cvtColor(photoMat, mGray, Imgproc.COLOR_BGR2GRAY, 3);
            loadCascadeFile(2);
            Detect(mGray);
            photoPath = "";
            break;
         default:
            break;
         }
      }

      @SuppressLint("NewApi")
      @Override
      protected void onActivityResult(int requestCode, int resultCode, Intent
   data) {
         // TODO Auto-generated method stub
         //super.onActivityResult(requestCode, resultCode, data);
         if(requestCode == pickCode && data!=null){
            mUri = data.getData();
            ivDisplay.setImageURI(mUri);
            btDetect.setVisibility(View.VISIBLE);

         }
         if(requestCode == captureCode && resultCode == RESULT_OK){
            ivDisplay.setImageURI(mUri);
            btDetect.setVisibility(View.VISIBLE);
         }
      }

   }

   package com.my.myapplication;

   import java.util.HashMap;
   import java.util.Map;

   public class Sign {
      public static final Map<Integer, Integer> myMap = new
   HashMap<Integer, Integer>() {{

      }};

      static{
         myMap.put(0, R.drawable.s0);
         myMap.put(1, R.drawable.s1);
         myMap.put(2, R.drawable.s2);
         myMap.put(3, R.drawable.s3);
         myMap.put(4, R.drawable.s4);
         myMap.put(5, R.drawable.s5);
         myMap.put(6, R.drawable.s6);
         myMap.put(7, R.drawable.s7);
         myMap.put(8, R.drawable.s8);
         myMap.put(9, R.drawable.s9);
         myMap.put(10, R.drawable.s10);
```

```
        myMap.put(11, R.drawable.s11);
        myMap.put(12, R.drawable.s12);
        myMap.put(13, R.drawable.s13);
        myMap.put(14, R.drawable.s14);
        myMap.put(15, R.drawable.s15);
        myMap.put(16, R.drawable.s16);
        myMap.put(17, R.drawable.s17);
        myMap.put(18, R.drawable.s18);
        myMap.put(19, R.drawable.s19);
        myMap.put(20, R.drawable.s20);
        myMap.put(21, R.drawable.s21);
        myMap.put(22, R.drawable.s22);
        myMap.put(23, R.drawable.s23);
        myMap.put(24, R.drawable.s24);
        myMap.put(25, R.drawable.s25);
    }
    private Integer id;
    private Float score;

    public Sign(Integer id, Float score) {
        super();
        this.id = id;
        this.score = score;
    }

    public Sign() {

    }

    public Integer getImage() {
        return myMap.get(id);
    }

    public Integer getId() {
        return id;
    }

    public Float getScore() {
        return score;
    }

    public void setScore(Float score) {
        this.score = score;
    }

}


package com.my.myapplication;

import android.app.Activity;
import android.content.Intent;
```

```
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;


public class MainActivity extends Activity implements OnClickListener
{
   private Button btRuntime;
   private Button btPickPhoto;
   private Button btTakePhoto;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      Initialize();
      btRuntime.setOnClickListener(this);
      //btTakePhoto.setOnClickListener(this);
      btPickPhoto.setOnClickListener(this);
   }
   public void Initialize(){
      btRuntime = (Button)findViewById(R.id.btRuntime);
      btPickPhoto = (Button)findViewById(R.id.btPickPhoto);
      //btTakePhoto = (Button)findViewById(R.id.btTakePhoto);
   }
   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      // Inflate the menu; this adds items to the action bar if it is
present.
      getMenuInflater().inflate(R.menu.main, menu);
//    MenuItem mi = menu.add(0, 1, 0, "Preferences");
//    mi.setIntent(new Intent(this, PrefActivity.class));
//    return super.onCreateOptionsMenu(menu);
      return true;
   }

   @Override
   public boolean onOptionsItemSelected(MenuItem item) {
      switch (item.getItemId()) {
         case R.id.action_settings:
            startActivity(new Intent(this, PrefActivity.class));
            return true;
         default:
            return super.onOptionsItemSelected(item);
      }
   }

   @Override
   public void onClick(View v) {
```

```
      // TODO Auto-generated method stub
      switch (v.getId()) {
      case R.id.btRuntime:
         Intent runtimeIntent = new Intent(MainActivity.this,
CameraActivity.class);
         startActivity(runtimeIntent);
         return;
      //case R.id.btTakePhoto:
         //Intent takePhotoIntent = new Intent(MainActivity.this,
TakePhotoActivity.class);
         //startActivity(takePhotoIntent);
         //return;
      case R.id.btPickPhoto:
         Intent pickPhotoIntent = new Intent(MainActivity.this,
PhotoActivity.class);
         startActivity(pickPhotoIntent);
         break;
      default:
         break;
      }
   }

}


import org.opencv.core.Point;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.core.Size;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.CLAHE;
import org.opencv.imgproc.Imgproc;
import org.opencv.objdetect.CascadeClassifier;

import java.io.File;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.logging.Logger;


public class CameraActivity extends Activity implements
CvCameraViewListener2 {

   private static final Logger logger =
Logger.getLogger(CameraActivity.class.toString());

   private static final Scalar FACE_RECT_COLOR     = new Scalar(0,
255, 0, 255);

   static {
      System.loadLibrary("caffe");
      System.loadLibrary("caffe_jni");
```

```
    }

    CLAHE clahe;
    int left, top, right, bottom;
    private CameraBridgeViewBase mCameraView;
    private ListView listDetectedSigns;
    private RelativeLayout listRelativeLayout;
    private CascadeClassifier cascadeClassifier;
    private ArrayList<Sign> listSign;
    private Detector detector;
    private CaffeMobile caffeMobile;
    private boolean change;
    private BaseLoaderCallback mLoaderCallback = new
BaseLoaderCallback(this) {
            @Override
            public void onManagerConnected(int status) {
                switch (status) {
                    case LoaderCallbackInterface.SUCCESS:
                       mCameraView.enableView();
                       detector = new Detector(CameraActivity.this);
                   clahe = Imgproc.createCLAHE();
                   caffeMobile = new CaffeMobile();
                   caffeMobile.loadModel("/sdcard/caffe_mobile/traffic-
signs/config_deploy.prototxt",
                           "/sdcard/caffe_mobile/traffic-
signs/_iter_10880.caffemodel");
                   caffeMobile.setScale(0.00390625F);
                        break;
                    default:
                        super.onManagerConnected(status);
                        break;
                }
            }
        };
    private Mat mRgba;
    private Mat mGray;

       //detector = new Detector(CameraActivity.this);
    private void Initialze(){
       mCameraView =
(CameraBridgeViewBase)findViewById(R.id.mCameraView);
       listDetectedSigns = (ListView)findViewById(R.id.listView1);
       listRelativeLayout =
(RelativeLayout)findViewById(R.id.listViewLayout);
       mCameraView.setCvCameraViewListener(this);
       listRelativeLayout.setVisibility(View.GONE);
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
```

```
getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
      setContentView(R.layout.camera_preview);
      Initialze();
      PreferenceManager.setDefaultValues(this, R.xml.pref, false);
   }

   @Override
    public void onResume() {
        super.onResume();
      if (!OpenCVLoader.initDebug()) {
            Log.d("fds", "Internal OpenCV library not found. Using
OpenCV Manager for initialization");
         OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_4_0,
this, mLoaderCallback);
        } else {
            Log.d("fds", "OpenCV library found inside package. Using
it!");

mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
        }

    }

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      // Inflate the menu; this adds items to the action bar if it is
present.
      getMenuInflater().inflate(R.menu.main, menu);
      return true;
   }

   @Override
   public void onCameraViewStarted(int width, int height) {
      // TODO Auto-generated method stub

   }

   @Override
   public void onCameraViewStopped() {
      // TODO Auto-generated method stub

   }

   @Override
   public Mat onCameraFrame(CvCameraViewFrame inputFrame) {
      //TODO Auto-generated method stub
      mRgba = inputFrame.rgba();

      Size sizeRgba = mRgba.size();
      int rows = (int) sizeRgba.height;
```

```
        int cols = (int) sizeRgba.width;

        left = cols / 2;
        top = 0;
        right = cols;
        bottom = rows * 3 / 4;

        mGray = inputFrame.gray();
        Log.i("STEP","step");
        mGray = mGray.submat(top, bottom, left, right);


        //Imgproc.equalizeHist(mGray, mGray);
//      CLAHE clahe = Imgproc.createCLAHE();
//      clahe.apply(mGray, mGray);
        if (!change) {
           MatOfRect signs = new MatOfRect();
           listSign = new ArrayList<Sign>();
           Log.i("STEP", "DetectBegin");
           detector.Detect(mGray, signs, 1);
           Log.i("STEP", "DetectEnd");
           Rect[] prohibitionArray = signs.toArray();
           Draw(prohibitionArray);
           change = true;
        } else{
           MatOfRect signs = new MatOfRect();
             detector.Detect(mGray, signs, 2);
             Rect[] dangerArray = signs.toArray();
             Draw(dangerArray);
           change = false;
        }


          //Core.rectangle(inputFrame, facesArray[i].tl(),
facesArray[i].br(), new Scalar(0, 255, 0, 255), 3);*/

        Imgproc.rectangle(mRgba, new Point(left, top), new Point(right,
bottom), FACE_RECT_COLOR, 2);
          return mRgba;
     }

     public void Draw(Rect[] facesArray){
//      if(facesArray.length<=0){
//          runOnUiThread(new Runnable() {
//
//           @Override
//           public void run() {
//              // TODO Auto-generated method stub
//              listRelativeLayout.setVisibility(View.GONE);
//           }
//       });
```

```
//
//        }
      //Imgproc.rectangle(mRgba,new Point(10, 10), new Point(100,
100), FACE_RECT_COLOR, 2);
        for (int i = 0; i <facesArray.length; i++){
          final int ii = i;
         Log.e("RES", "start");
          Mat subMat;
          subMat = mGray.submat(facesArray[i]);
         Mat resizeMat = new Mat();
         Imgproc.resize(subMat, resizeMat, new Size(32, 32), 0, 0,
Imgproc.INTER_CUBIC);

         clahe.apply(resizeMat, resizeMat);

         //Core.flip(resizeMat.t(), resizeMat, 0);
         resizeMat = resizeMat.t();
         File fileDir = getImageFile();
         Imgcodecs.imwrite(fileDir.toString(), resizeMat);
         Log.e("RES", "start2");
         final int[] result =
caffeMobile.predictImage(Uri.fromFile(fileDir).getPath());
         Log.e("RES", "result:" + Arrays.toString(result));
         final float[] rr =
caffeMobile.getConfidenceScore(Uri.fromFile(fileDir).getPath());
         Log.e("RES", "rr:" + Arrays.toString(rr));
         Mat mat = Imgcodecs.imread(Uri.fromFile(fileDir).getPath(), -
1);
         //Sign.myMap.put("image"+result[0],
Utilities.convertMatToBitmap(mat));


         if (fileDir.exists()) {
            if (fileDir.delete()) {
               System.out.println("file Deleted :" + fileDir);
            } else {
               System.out.println("file not Deleted :" + fileDir);
            }
         }
         Log.e("RES", "stop");
         Point tl = facesArray[i].tl();
         tl.set(new double[]{tl.x + left, tl.y + top});
         Point br = facesArray[i].br();
         br.set(new double[]{br.x + left, br.y + top});

         Imgproc.rectangle(mRgba, tl, br, FACE_RECT_COLOR, 2);
           //if (rr[result[0]] > 0.9) {
         // runOnUiThread(new Runnable() {

//             @Override
//             public void run() {
```

```
//                // TODO Auto-generated method stub
//                Sign sign = new Sign(result[0], rr[result[0]]);
//                listSign.add(sign);
//                listRelativeLayout.setVisibility(View.VISIBLE);
//                itemAdapter adapter = new itemAdapter(listSign,
CameraActivity.this);
//                adapter.notifyDataSetChanged();
//                listDetectedSigns.setAdapter(adapter);
//             }
//          });
   //    }

         }
   }

   public File getImageFile() {
      File mediaStorageDir = new
File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTO
RY_PICTURES), "Signs");

      if (!mediaStorageDir.exists()) {
         if (!mediaStorageDir.mkdirs()) {
            Log.e("G", "failed to create directory");
            return null;
         }
      }

      //String timeStamp = new
SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
      File mediaFile;
      mediaFile = new File(mediaStorageDir.getPath() + File.separator
+ "testimage"  + ".jpg");
      //mediaFile = new File(mediaStorageDir.getPath() +
File.separator + "testimage_20160425_175554.jpg");
      return mediaFile;
   }
}
package com.my.myapplication;

import android.app.Activity;
import android.content.Context;
import android.content.SharedPreferences;
import android.preference.PreferenceManager;
import android.util.Log;

import org.opencv.core.Mat;
import org.opencv.core.MatOfRect;
import org.opencv.core.Size;
import org.opencv.objdetect.CascadeClassifier;

import java.io.File;
```

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.logging.Logger;

public class      Detector {

   private static final Logger logger =
Logger.getLogger(Detector.class.getName());
   Integer minSize1;
   Integer maxSize1;
   Integer minSize2, maxSize2;
   SharedPreferences sp;
   private Activity activity;
   private CascadeClassifier cascadeClassifier1;
   private CascadeClassifier cascadeClassifier2;
   public Detector(Activity activity){
      this.activity = activity;
      loadCascadeFile(1);
      loadCascadeFile(2);
      sp = PreferenceManager.getDefaultSharedPreferences(activity);
      minSize1 = Integer.parseInt(sp.getString("minSize1", "30"));
      maxSize1 = Integer.parseInt(sp.getString("maxSize1", "70"));
      minSize2 = Integer.parseInt(sp.getString("minSize2", "30"));
      maxSize2 = Integer.parseInt(sp.getString("maxSize2", "70"));

   }
   public void Detect(Mat mGray,MatOfRect signs,int type){
      //loadCascadeFile(type, cascadeClassifier);
//    loadCascadeFile(type);
      switch (type) {
         case 1:
            if (cascadeClassifier1 != null &&
!cascadeClassifier1.empty()) {
               cascadeClassifier1.detectMultiScale(mGray, signs, 1.1,
3, 0
                     , new Size(minSize1, minSize1), new
Size(maxSize1, maxSize1));
            } else {
               Log.e("s", "cascade");
            }
            break;
         case 2:
         default:
            if (cascadeClassifier2 != null &&
!cascadeClassifier2.empty()) {
               cascadeClassifier2.detectMultiScale(mGray, signs, 1.1,
5, 0
                     , new Size(minSize2, minSize2), new
Size(maxSize2, maxSize2));
            } else {
```

```
                Log.e("s", "cascade");
            }
        }
    }
    private void loadCascadeFile(int type){
        try {
            InputStream is;
            File cascadeDir = activity.getDir("cascade",
Context.MODE_PRIVATE);
            File cascadeFile;
            switch (type) {
            case 1:
                is =
activity.getResources().openRawResource(R.raw.circle);

                cascadeFile = new File(cascadeDir, "circle.xml");
                break;
            case 2:
            default:
                is =
activity.getResources().openRawResource(R.raw.triangle);

                cascadeFile = new File(cascadeDir, "triangle.xml");
                break;
            }

            FileOutputStream os = new FileOutputStream(cascadeFile);
            byte[] buffer = new byte[4096];
                int bytesRead;
                while ((bytesRead = is.read(buffer)) != -1) {
                    os.write(buffer, 0, bytesRead);
                }
                is.close();
                os.close();

                // Load the cascade classifier
            switch (type) {
                case 1:
                    cascadeClassifier1 = new
CascadeClassifier(cascadeFile.getAbsolutePath());
                    break;
                case 2:
                default:
                    cascadeClassifier2 = new
CascadeClassifier(cascadeFile.getAbsolutePath());
                    break;
            }

        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
```

```
      }
   }
}
package com.my.myapplication;

import android.annotation.SuppressLint;
import android.app.Activity;
import android.database.Cursor;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Environment;
import android.provider.MediaStore;
import android.util.Log;

import org.opencv.android.Utils;
import org.opencv.core.Mat;

import java.io.File;
import java.text.SimpleDateFormat;
import java.util.Date;

public class Utilities {
   public static final int MEDIA_TYPE_IMAGE = 1;
   public static final int MEDIA_TYPE_VIDEO = 2;

   public static Bitmap convertMatToBitmap(Mat src){
      Bitmap bm = Bitmap.createBitmap(src.cols(),
            src.rows(), Bitmap.Config.ARGB_8888);
        Utils.matToBitmap(src, bm);
        //Imgproc.cvtColor(src, dst, code, dstCn)
        return bm;
   }

   public static String getRealPathFromURI(Uri contentURI,Activity
activity) {
       String path;
       Cursor cursor = activity.getContentResolver().query(contentURI,
null, null, null, null);
       if (cursor == null) { // Source is Dropbox or other similar
local file path
           path = contentURI.getPath();
       } else {
           cursor.moveToFirst();
           int idx =
cursor.getColumnIndex(MediaStore.Images.ImageColumns.DATA);
           path = cursor.getString(idx);
           cursor.close();
       }
       return path;
   }
```

```
    public static Uri getOutputMediaFileUri(int type){
          return Uri.fromFile(getOutputMediaFile(type));
    }
    /** Create a File for saving an image or video */
    @SuppressLint("SimpleDateFormat")
    public static File getOutputMediaFile(int type){
        // To be safe, you should check that the SDCard is mounted
        // using Environment.getExternalStorageState() before doing
this.
      File mediaStorageDir = new
File(Environment.getExternalStoragePublicDirectory(
            Environment.DIRECTORY_PICTURES),
"TrafficSignsRecognition");
        // This location works best if you want the created images to
be shared
        // between applications and persist after your app has been
uninstalled.
        // Create the storage directory if it does not exist
        if (! mediaStorageDir.exists()){
            if (! mediaStorageDir.mkdirs()){
                Log.d("TrafficSignsRecognition", "failed to create
directory");
                return null;
            }
        }

        // Create a media file name
        String timeStamp = new
SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
        File mediaFile;
        if (type == MEDIA_TYPE_IMAGE){
            mediaFile = new File(mediaStorageDir.getPath() +
File.separator +
            "IMG_"+ timeStamp + ".jpg");
        } else if(type == MEDIA_TYPE_VIDEO) {
            mediaFile = new File(mediaStorageDir.getPath() +
File.separator +
            "VID_"+ timeStamp + ".mp4");
        } else {
            return null;
        }

        return mediaFile;
    }

}
```